ГУАП

КАФЕДРА № 44

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доц., канд. техн. наук, доц.		О.О. Жаринов
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №9

РАЗРАБОТКА МИКРОПОЦЕССОРНОГО ИНДИКАТОРА ВЕЛИЧИНЫ НАПРЯЖЕНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4143		Е.Д.Тегай
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

Цель работы

Изучить методы схемотехнической и программной реализации интерфейсного взаимодействия модулей вычислительных устройств и аналого-цифровых преобразователей. Разработать модель микропроцессорного индикатора величины электрического напряжения и осуществить моделирование его работы.

Вариант задания

Содержание индивидуального задания №9 продемонстрировано на рисунке 1. Для удобства необходимые данные и номер варианта выделены жёлтым цветом.

Вар.	1	2	3	4	5	6	7	8	9	10
АЦП	AD1674	ADS7825	LTC1099	LTC1292	LTC1860	MAX1240	MCP3001	MCP3221	TLC548	ADC0801

Рисунок 1 – Вариант АЦП

Описание концепции проектирования

Данный проект представляет собой схему, состоящую из нескольких ключевых компонентов. В основе схемы лежит микроконтроллер Arduino 328, который выступает в качестве центрального управляющего устройства. Для преобразования аналоговых сигналов в цифровую форму используется АЦП серии TLC548. ЦАП AD3510 выполняет обратную функцию, преобразуя цифровые сигналы, полученные от микроконтроллера, обратно в аналоговую форму. Полученные данные выводятся на восьми индикаторах, которые реагируют на значения в определённых диапазонах возможных значений АЦП.

Результат работы демонстрируется на осциллографе в первых двух каналах. На канал А подаётся исходный сигнал (например, синусоида). На канал В подаётся результат работы обоих объектов – АЦП и ЦАП. В случае, если всё правильно настроено и подключено, то сигналы должны быть идентичны друг другу.

Описание основных характеристик модели АЦП

В разработанной схеме используется АЦП серии TLC548. Он представляет собой 8-битное устройство, обладающее широким диапазоном входных напряжений, который составляет от 0 до напряжения источника питания (в данном случае это будет 5В). Время преобразования составляет 17 мкс максимум.

Данный АЦП обладает высокой точностью и стабильностью преобразования, а также низкий уровень шума и энергопотребления.

Временная диаграмма интерфейса АЦП

Искомая временная диаграмма продемонстрирована на рисунке 2.

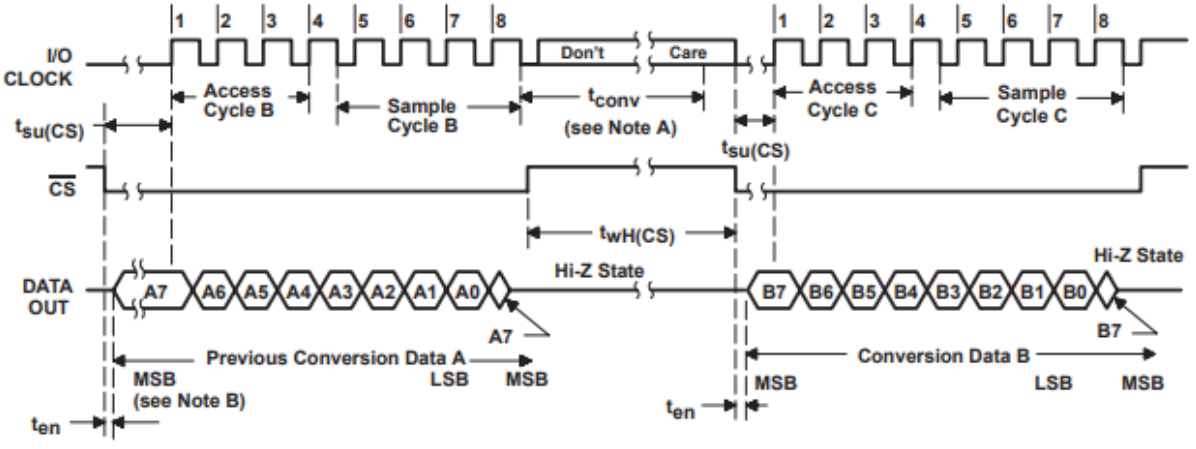

Рисунок 2 – Временная диаграмма АЦП

Как видно из рисунка 2, на временной диаграмме присутствуют 3 линии. Разберём каждую из них.

I/O CLOCK. Это сигнал тактирования, который управляет обменом данными между микроконтроллером и АЦП. Он указывает моменты времени, когда данные передаются между устройствами. На сигнале, собственно, и выделены участки, когда идёт приём и сэмплирование аналогового сигнала. Также виден участок времени (t_{conv}), в период которого идёт время преобразования (из аналога в цифру).

CS. Это сигнал выбора микросхемы. Он указывает микроконтроллеру, что данный АЦП выбран для обмена данными. Когда CS активен, микроконтроллер и АЦП могут обмениваться данными. Также можно заметить участок времени (t_{wH}), который представляет собой время установки

уровня высокого сигнала. Это время обозначает время установки стабильного высокого уровня после активации сигнала.

DATA OUT. Это сигнал, который содержит цифровые данные, выходящие из АЦП.

Также глобально на диаграмме видны некоторые диапазоны (данных или времени). Так, например, MSB – это участок времени, который указывает на момент передачи самого значимого (старшего) бита данных по интерфейсу SPI. Аналогичен смысл и LSB, только это касается самого младшего бита. t_{en} – это участок времени, который отведён для времени управления (время, в течение которого данные на выходе АЦП остаются стабильными и готовыми для чтения после активации CS).

Схема в среде Proteus

Разработанная схема продемонстрирована на рисунке 3.

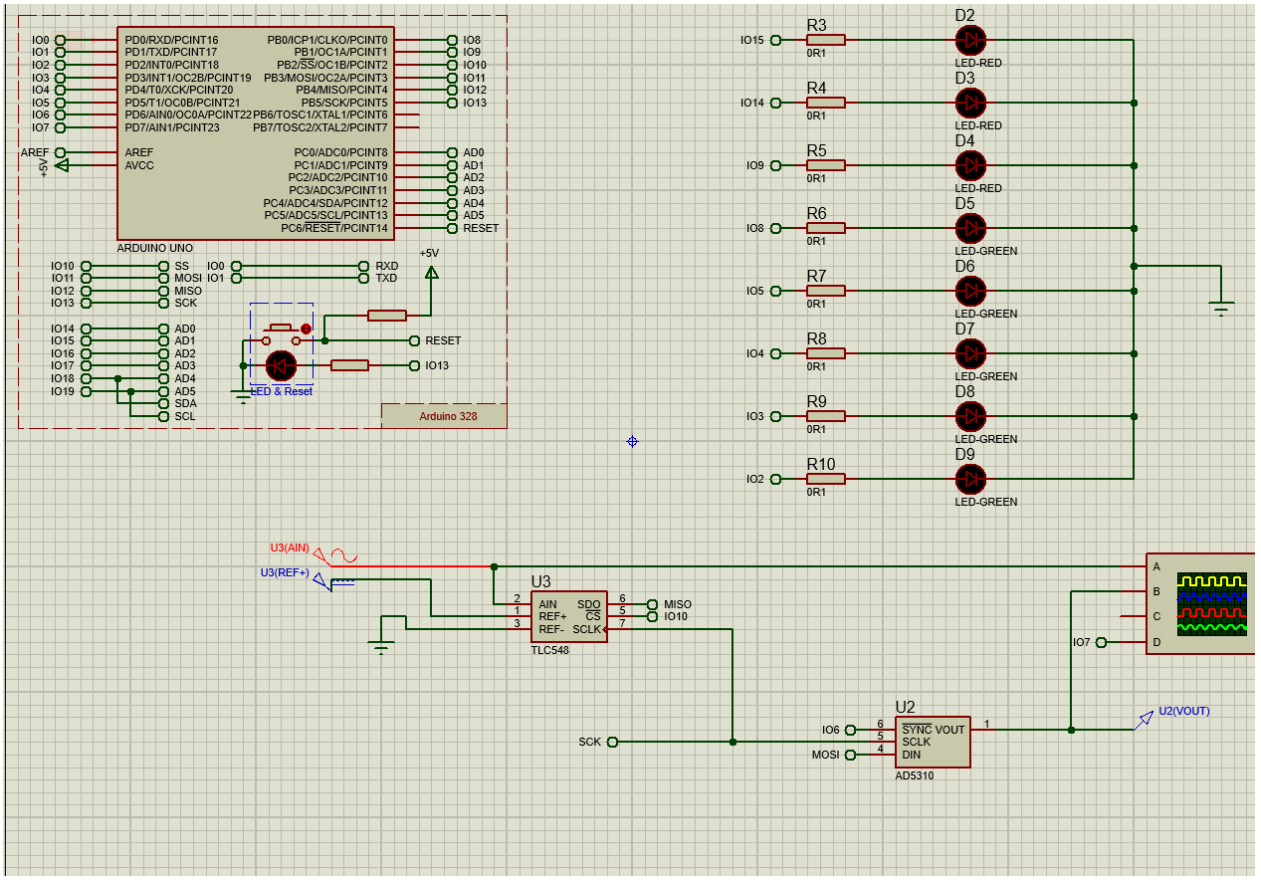

Рисунок 3 – Схема

Блок-схема программы

Искомая блок-схема программы показана на рисунке 4.

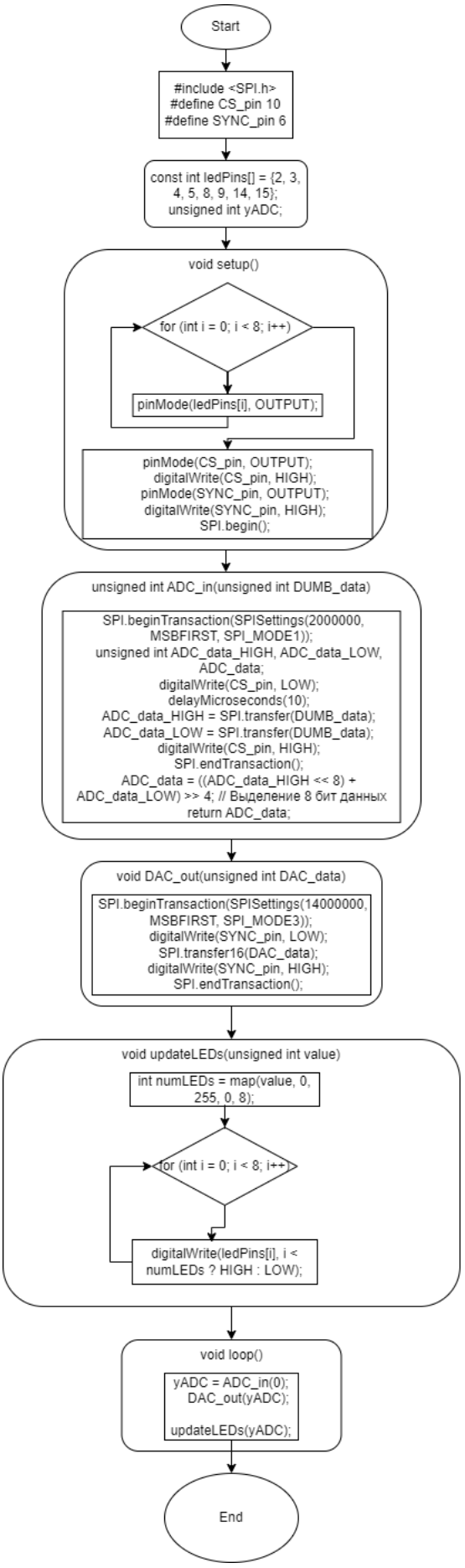

Рисунок 4 – Блок-схема

Текст программы прошивки управляющего микроконтроллера с комментариями

Перед непосредственным листингом программы более подробно рассмотрим разработанную программу.

В качестве языка программирования был выбран C++. Программа используется для работы с выбранным микроконтроллером Arduino и включает в себя библиотеку для работы с интерфейсом SPI. В начале программе идёт, собственно, импорт этой библиотеки, а также идёт определение пинов, которые будут использоваться для управления SPI с устройствами.

Далее идёт объявление массива для хранения номеров пинов, к которым подключены светодиоды, а также переменную для хранения данных, полученных от АЦП.

Затем идёт объявление нескольких функций. Разберём каждую из них.

- **Setup().** Эта функция выполняется один раз при старте программы (или при сбросе микроконтроллера). Она используется для инициализации настроек и подготовительных действий. Цикл внутри неё инициализирует все пины, указанные в объявленном ранее массиве, как выходные.

Далее идёт установка пина CS в режим OUTPUT (он используется для управления сигналом выбора микросхемы). Затем идёт установка сигнала на пине CS в высокий уровень. Это отключает АЦП, пока не потребуется чтение данных. После этого идёт установка пина SYNC в режим OUTPUT. Он используется для управления сигналом синхронизации АЦП. Затем идёт установка сигнала на пине SYNC в высокий уровень. Это отключает ЦАП, пока не потребуется запись данных. Завершающим действием является инициализация интерфейса SPI, а именно его включение и настройка соответствующих пинов (MISO, MOSI, SCK) для работы в режиме SPI.

- **ADC_in.** Данная функция принимает одно входное значение. Предназначена для чтения данных с АЦП. После этого идёт начало транзакции SPI с настройками:

Частота 2 МГц

Передача старшего бита вперёд (MSB First)

Режим SPI = 1

Далее идёт объявление переменных, которые потом будут использоваться для хранения данных, полученных от АЦП. Затем устанавливается сигнал на пине CS в низкий уровень, активируя АЦП для начала передачи данных. Потом идёт небольшая задержка в 10 мкс для стабилизации сигнала после активации АЦП.

Затем идёт отправка «тупых» данных по SPI и одновременное получение старшего бита данных от АЦП, который сохраняется в ADC_data_HIGH. Далее идёт повторная отправка «тупых» данных с получением уже младшего бита, который сохраняется в соответствующую переменную ADC_data_LOW. После завершения транзакции идёт выделение 8 значащих битов данных и возвращение результата преобразования АЦП.

- **DAC_out.** Эта функция принимает одно входное значение и ничего не возвращает на выходе. Она предназначена лишь для отправки данных на ЦАП. Внутри функции идёт начало транзакции SPI с настройками:

Частота 14 МГц

Передача старшего бита вперёд (MSB First)

Режим SPI 3

Затем идёт установка сигнала на пине SYNC в низкий уровень, активируя ЦАП для начала передачи данных. После этого идёт отправка данных по SPI на ЦАП, а далее – установка сигнала на пине SYNC в высокий уровень, деактивируя ЦАП после завершения передачи данных. На этом и завершается текущая транзакция SPI.

- **updateLEDs.** Эта функция принимает одно входное значение и ничего не возвращает на выходе. Она нужна для обновления состояния светодиодов на основе входного значения. Далее с помощью функции map идёт преобразования значения из диапазона 0-255 в диапазон 0-8. Это значение определяет, сколько светодиодов должно быть включено. Чтобы пройтись по

всем светодиодам, используется соответствующий цикл. Затем идёт установка состояния каждого светодиода в зависимости от текущего значения.

- **loop**. Это основная функция в Arduino. Она выполняется непрерывно после setup(). Внутри неё сначала идёт вызов функции ADC_in с аргументом 0 и присваивание возвращенного значения переменной yADC. После этого функция DAC_out отправляет значение yADC на ЦАП, чтобы преобразовать цифровое значение обратно в аналоговый сигнал. Завершается всё функцией updateLEDs, которая обновляет состояние светодиодов на основе значения yADC, полученного от АЦП.

Листинг программы продемонстрирован ниже.

Листинг программы

```
#include <SPI.h>
#define CS_pin 10
#define SYNC_pin 6

const int ledPins[] = {2, 3, 4, 5, 8, 9, 14, 15};
unsigned int yADC;

void setup() {
   for (int i = 0; i < 8; i++) {
      pinMode(ledPins[i], OUTPUT);
   }
   pinMode(CS_pin, OUTPUT);
   digitalWrite(CS_pin, HIGH);
   pinMode(SYNC_pin, OUTPUT);
   digitalWrite(SYNC_pin, HIGH);
   SPI.begin();
}

unsigned int ADC_in(unsigned int DUMB_data) {
   SPI.beginTransaction(SPISettings(2000000, MSBFIRST, SPI_MODE1));
   unsigned int ADC_data_HIGH, ADC_data_LOW, ADC_data;
   digitalWrite(CS_pin, LOW);
   delayMicroseconds(10);
   ADC_data_HIGH = SPI.transfer(DUMB_data);
   ADC_data_LOW = SPI.transfer(DUMB_data);
   digitalWrite(CS_pin, HIGH);
   SPI.endTransaction();
   ADC_data = ((ADC_data_HIGH << 8) + ADC_data_LOW) >> 4; // Выделение
```

```
8 бит данных
   return ADC_data;
}

void DAC_out(unsigned int DAC_data) {
   SPI.beginTransaction(SPISettings(14000000, MSBFIRST, SPI_MODE3));
   digitalWrite(SYNC_pin, LOW);
   SPI.transfer16(DAC_data);
   digitalWrite(SYNC_pin, HIGH);
   SPI.endTransaction();
}

void updateLEDs(unsigned int value) {
   int numLEDs = map(value, 0, 255, 0, 8); // Преобразование значения АЦП в
количество светодиодов
   for (int i = 0; i < 8; i++) {
      digitalWrite(ledPins[i], i < numLEDs ? HIGH : LOW);
   }
}

void loop() {
   yADC = ADC_in(0);
   DAC_out(yADC);
   updateLEDs(yADC);
}
```

Временная диаграмма (осциллограмма)

Так как на схеме использовался генератор сигнала, то можно провести различные тесты в зависимости от исходного сигнала. Так, итоговая работа относительно синусоидального сигнала продемонстрирована на рисунке 6, а на рисунке 5 показаны характеристики сигнала. Аналогично на рисунках 7 – 8 изображены характеристики и результат относительно прямоугольного импульсного сигнала.

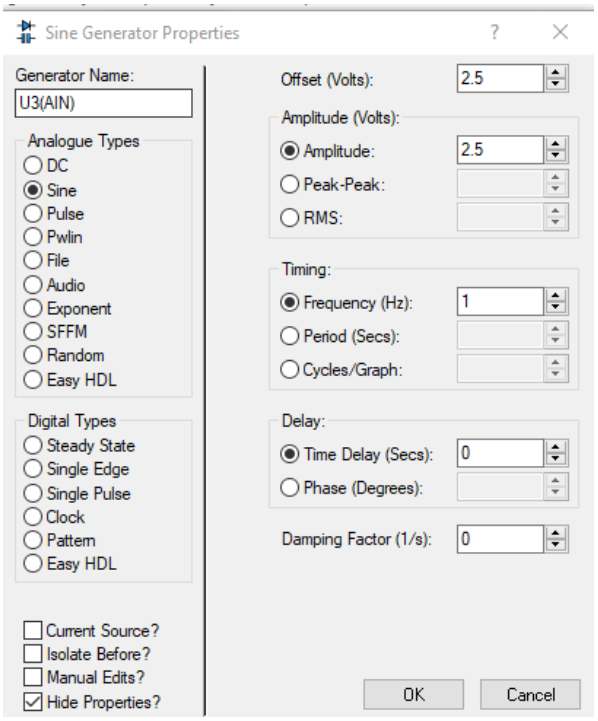

Рисунок 5 – Характеристики сигнала

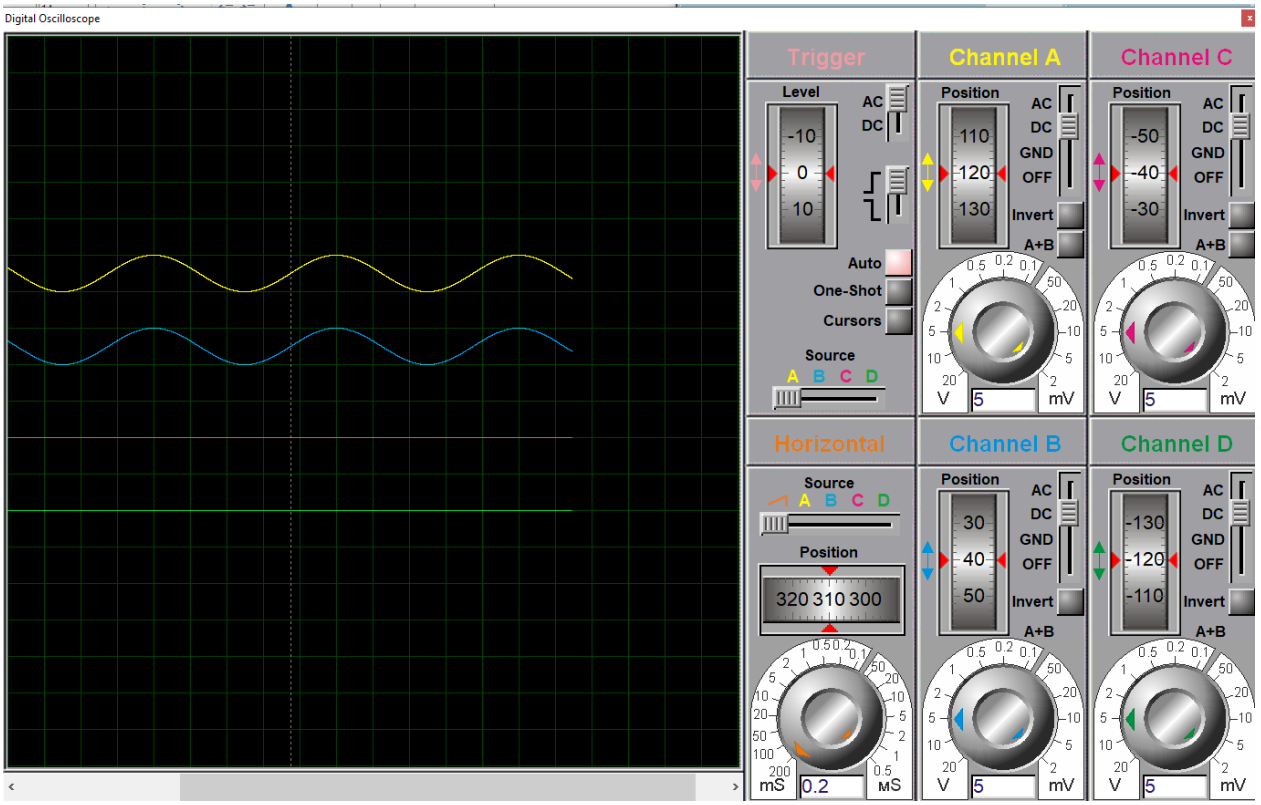

Рисунок 6 – Синусоидальный сигнал

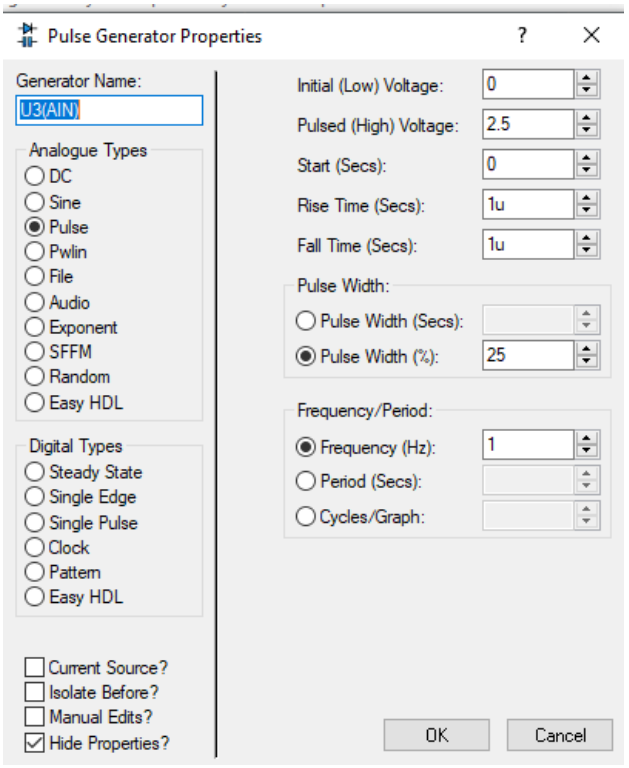

Рисунок 7 – Характеристики сигнала

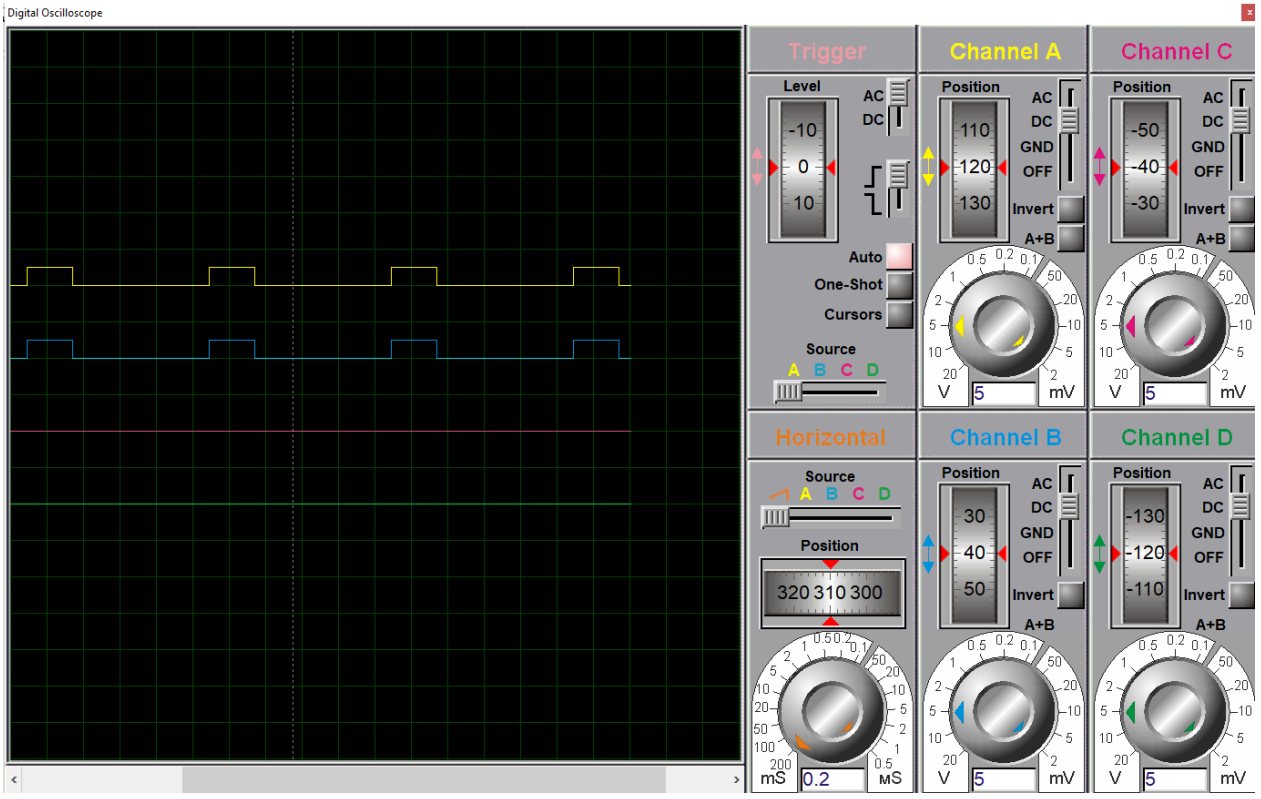

Рисунок 8 – Прямоугольные импульсы

Иллюстрации шкального индикатора

Искомые иллюстрации для первого и второго тестов продемонстрированы на рисунках 9 – 13.

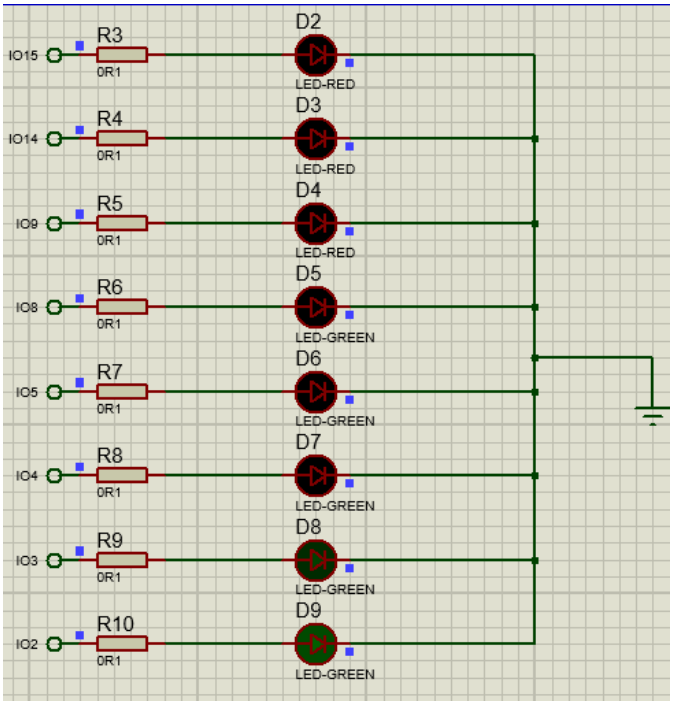

Рисунок 9 – Шкальный индикатор (синусоидальный сигнал, начало)

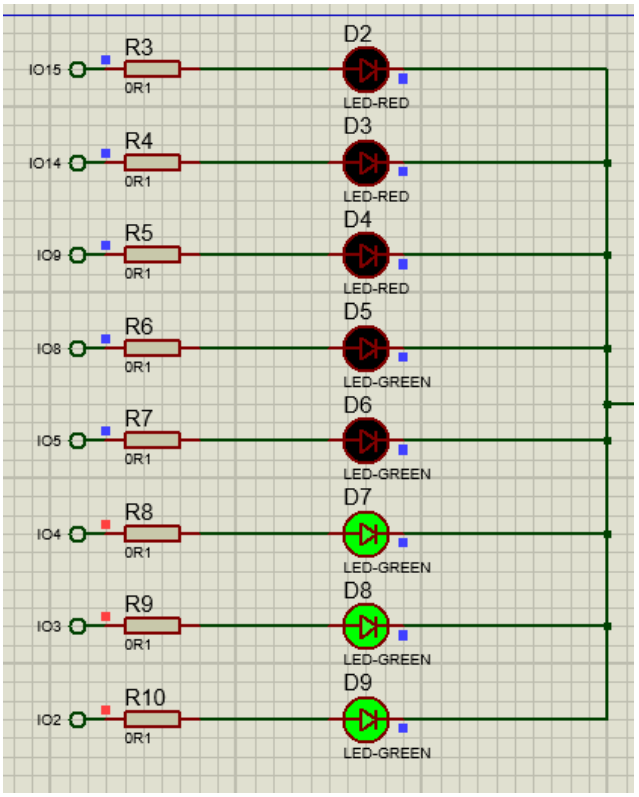

Рисунок 10 – Шкальный индикатор (синусоидальный сигнал, середина)

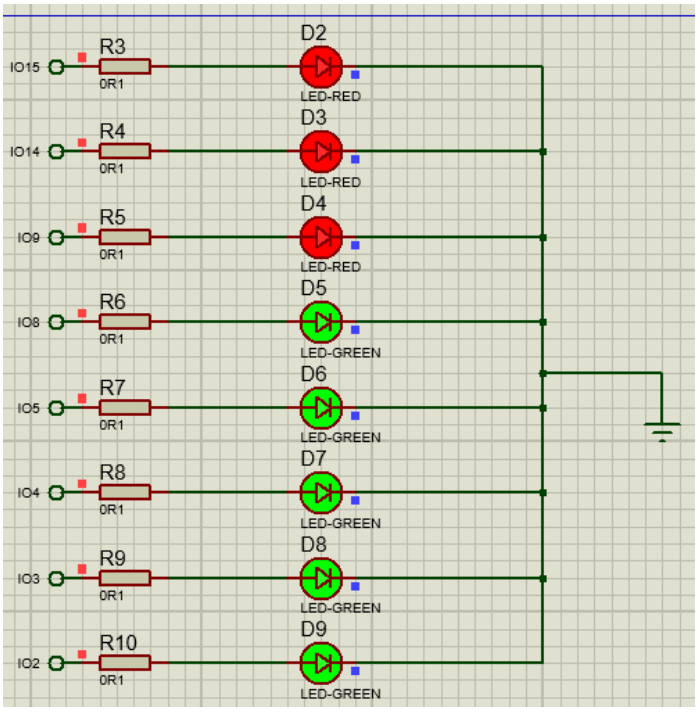

Рисунок 11 – Шкальный индикатор (синусоидальный сигнал, окончание)

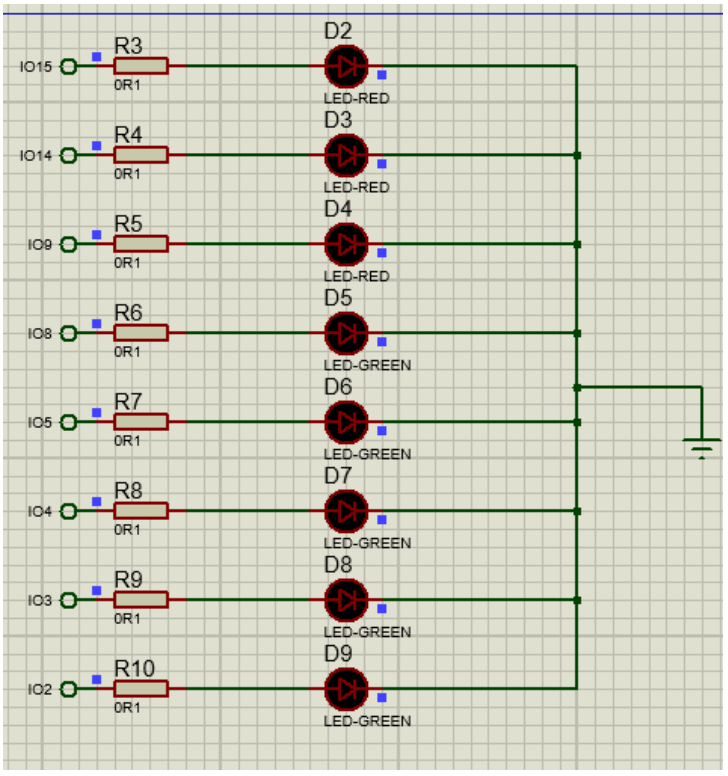

Рисунок 12 – Шкальный индикатор (прямоугольный сигнал, начало)

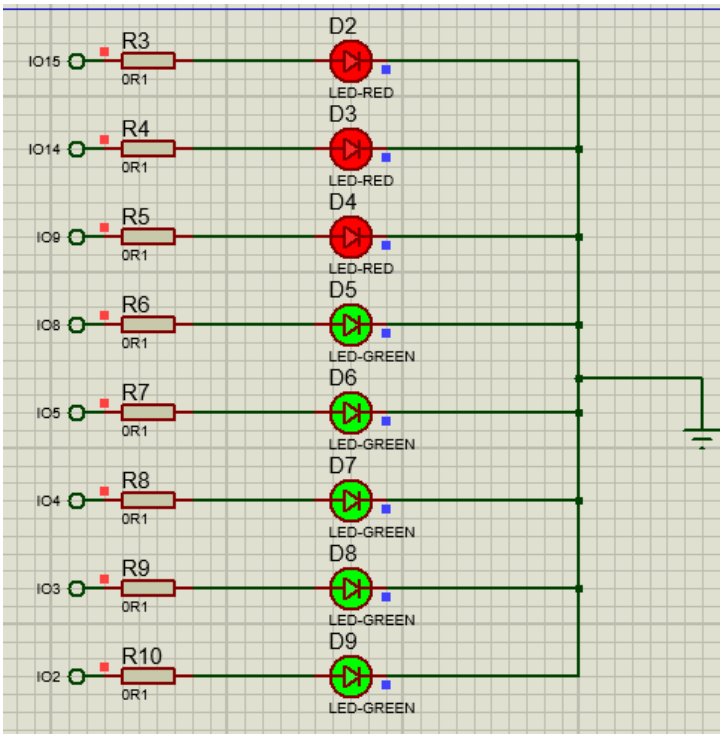

Рисунок 13 – Шкальный индикатор (прямоугольный сигнал, окончание)

Выводы

В данной лабораторной работе были изучены методы схемотехнической и программной реализации интерфейсного взаимодействия модулей вычислительных устройств и аналого-цифровых преобразователей. Разработана модель микропроцессорного индикатора величины электрического напряжения и осуществлена моделирование его работы.

Список используемых источников

1. Проектирование встраиваемых систем на ПЛИС. / З.Наваби; перев. с англ. В.В. Соловьева. – М.: ДМК Пресс, 2016. - 464 с.

2. Проектирование цифровых устройств на ПЛИС: учеб. пособие / И.В. Ушенина. - СПб: Лань, 2022. - 408 с.

3. Цифровая схемотехника и архитектура компьютера / Д.М. Харрис, С.Л. Харрис; пер. с англ. ImaginationTechnologies. – М.: ДМК Пресс, 2018. - 792 с.

4. Методические указания: [Электронный ресурс] //Санкт-Петербургский государственный университет аэрокосмического приборостроения.URL.:

https://pro.guap.ru/inside/student/tasks/a3c2a9a2c63d7a16036b6bc107abacdb/download. (Дата обращения: 26.05.24).

5. Лекция №6 от 22 апреля 2024 года: [Электронный ресурс] // Санкт-Петербургский государственный университет аэрокосмического приборостроения. URL.: https://bbb2.guap.ru/playback/presentation/2.3/98ebb27d1dba4a4bac6ecf053e0ae9024373948a-1713787146024. (Дата обращения: 26.05.24).

6. Лекция №7 от 6 мая 2024 года: [Электронный ресурс] // Санкт-Петербургский государственный университет аэрокосмического приборостроения. URL.: https://bbb2.guap.ru/playback/presentation/2.3/d4b490cf4cf7d7ae65c3b14b7d239ccb9e78f2a8-1714996255068. (Дата обращения: 26.05.24).

7. Официальная документация ЦАП серии AD5310: [Электронный ресурс]. URL.: https://www.farnell.com/datasheets/14097.pdf. (Дата обращения: 26.05.24).

8. Официальная документация АЦП серии TLC548: [Электронный ресурс]. URL.: https://www.datasheet.eicom.ru/T/tlc548.pdf. (Дата обращения: 26.05.24).

9. Схемотехника телекоммуникационных устройств: учебник / С.И. Зиатдинов, Т.А. Суетина, Н.В. Поваренкин. - М.: Академия, 2013. - 368 с. [Библиотечный шифр 621.396 З-59]

10. X51-совместимые микроконтроллеры фирмы Cygnal: монография / О.И.Николайчук. - М.: ИД СКИМЕН, 2002. - 471 с. [Библиотечный шифр 004 Н63]

11. Основы микропроцессорной техники: учебное пособие / П.Н. Неделин. - СПб: Изд-во ГУАП, 2013. - 63 с. [Библиотечный шифр 004.3(075) Н42]

12. Keil uVision. // URL: http://cxem.net/software/keil.php

13. Keil uVision3 V3.51 для платформы MCS51 // URL: http://microsin.net/programming/MSC51/keil-uvision3-v351.html

14. Проектирование электронных устройств в Proteus 8.1. Часть 12. // URL: http://cxem.net/comp/comp204.php

15. Система моделирования ISIS Proteus. Быстрый старт. // URL: http://easyelectronics.ru/sistema-modelirovaniya-isis-proteus-bystryj-start.html
16. Книжная полка: 5 лучших книг о платформе Arduino // URL: https://arduinoplus.ru/5-knig-ob-arduino/