

Міністерство освіти і науки України
Запорізький національний технічний університет

Кафедра ПЗ

Методичні вказівки

до самостійної роботи
з дисципліни

«Технології та системи віртуальної та віддаленої інженерії»
для студентів спеціальності
121 «Інженерія програмного забезпечення»
всіх форм навчання

2017

Методичні вказівки до самостійної роботи з дисципліни «Технології та системи віртуальної та віддаленої інженерії» для студентів спеціальності 121 «Інженерія програмного забезпечення» всіх форм навчання / Укл.: А.В. Пархоменко, О.М. Гладкова. – Запоріжжя: ЗНТУ, 2017. – 47 с.

Укладачі: А. В. Пархоменко, к.т.н., доцент кафедри ПЗ,
О. М. Гладкова асистент кафедри ПЗ.

Рецензент: Г. М. Шило, к.т.н., доцент кафедри ІТЕЗ.

Відповідальний

за випуск: С.О. Субботін, зав. кафедри ПЗ, д.т.н., професор

Затверджено
на засіданні кафедри
Програмних засобів

Протокол № 11
від 06.06.2017 р.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	4
Самостоятельная работа №1	
Реализация экспериментов в среде удаленной лаборатории RELDES	6
Самостоятельная работа № 2	
Реализация эксперимента «Плата быстрого прототипирования» в среде удаленной лаборатории GOLDI	25
Контрольные вопросы	46
ЛИТЕРАТУРА	47

ВВЕДЕНИЕ

В настоящее время задачи развития информационно-коммуникационных технологий в образовании являются приоритетными.

Наряду с традиционными технологиями образования, все большее распространение получает дистанционное образование и использованием передовых средств информационных технологий. Дистанционное образование помогает решать задачи обучения и повышения квалификации людей, которые не могут присутствовать на занятиях в учебном заведении.

С развитием цифровых технологий и Интернета стали использовать так называемые онлайн сервисы для разносторонних потребностей людей. Интернет намного упростил связь, необходимую для того, чтобы связать два компьютера или другие виды устройств в разных частях Земли. Это позволило воплотить в жизнь концепцию удаленного управления оборудованием реальных лабораторий. На начальном этапе своего развития данная концепция предусматривала только интеграцию в учебный процесс в технических вузах, в том числе и в системе дистанционного образования. Впоследствии технология развивалась и распространялась, поэтому сейчас благодаря Интернету студентам стал доступен коллективный доступ к одной удаленной лаборатории, а не по отдельности. Студенты получили доступ к большим экспериментальным установкам, при этом нет необходимости приобретать больше одной на всю отрасль для проведения практических занятий, что позволяет сэкономить средства. Также по специальным программам был предоставлен доступ к уникальному оборудованию, недоступного обычным пользователям.

Следует подчеркнуть также различие понятий удаленной лаборатории не только от обычной, где традиционным способом эксперимент проводится наряду с реальным оборудованием, но и с виртуальной. Удаленные лаборатория предоставляют больше преимуществ, чем виртуальные лаборатории, однако имеют с ними определенные характерные черты такие как: доступ на расстоянии с помощью сети Интернет, имитация панели управления прибором, датчиками и т.д. Но в виртуальной лаборатории все оборудование

заменено на симуляцию компьютерной программой. Таким образом вместо реального физического процесса виртуальная лаборатория позволяет исследовать лишь математическую модель физического явления. Поэтому виртуальная лаборатория используется исключительно для обучения или для подготовки к работе с реальным оборудованием.

Использование удаленных лабораторий имеют такие экономические преимущества:

- экономия средств на создание реальных экспериментальных установок при наличии их хотя бы в единичном экземпляре в любом реальном практикуме, а также экономия средств на создание реальных установок за счет отсутствия необходимости создавать реальный интерфейс для работы с аппаратурой в лаборатории, полностью заменяя его сообщением с компьютерным сервером;

- появление возможности работы на уникальном дорогостоящем оборудовании, которое находится в любом городе любой страны в крупных научных центрах;

- удобство доступа в лабораторию в любое время для любого количества пользователей, в том числе одновременно (если это возможно реализовать);

- возможность предложить услуги уникального оборудования для экспериментов любому желающему на коммерческой или иной основе, тем самым увеличивая КПД отдачи от уже существующей аппаратуры;

- применение современных достижений педагогики в компьютерной реализации удаленного доступа к традиционным экспериментам.

Поэтому концепция использования удаленных лабораторий является очень актуальной для нашей страны, это даст возможность делиться полезным опытом между странами, которые уже активно пользуются этими технологиями, где уже поняли, что использование такой формы практикума является не только очень информативным, но и стоит гораздо меньше, чем ранее при использовании реальных лабораторий везде.

Самостоятельная работа №1

Реализация экспериментов в среде удаленной лаборатории RELDES

1. Теоретические сведения

Главная страничка (рис.1.1) удаленной лаборатории (УЛ) RELDES содержит описание преимуществ и основных функциональных возможностей УЛ.

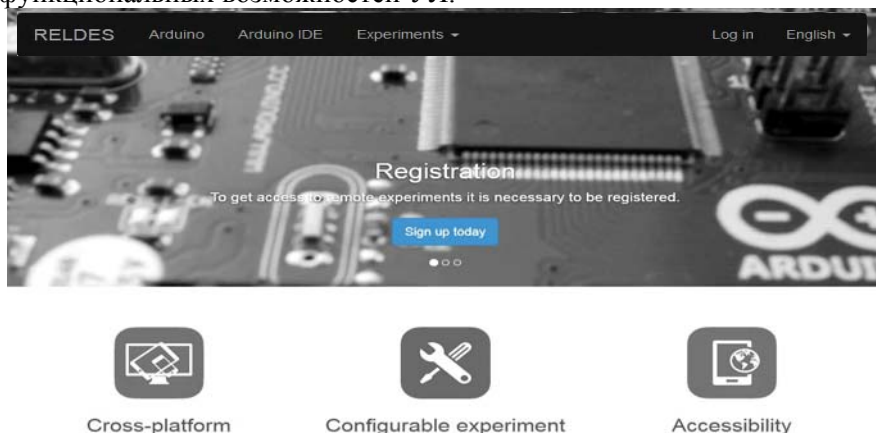


Рисунок 1.1 – Главная страница

В главном меню программы (рис.1.1) содержатся такие пункты:

- RELDES;
- Arduino;
- Arduino IDE;
- Experiments.

Также в главном меню есть кнопка регистрации «Log in» и изменения языка интерфейса.

Пункт меню «Experiments» содержит подменю (рис.1.2), которое включает в себя дополнительные страницы:

- Distance sensor;
- Servo;
- LED;
- Display;
- User manual.

Пункты подменю: Distance sensor; Servo; LED; Display — это 4 доступных эксперимента удаленной лаборатории.



Рисунок 1.2– Подменю главного меню «Experiment»

Страница эксперимента разделена на несколько блоков (рис.1.3). В верхней части страницы эксперимента под главным меню программы формируется ленточное меню, которое позволяет быстро переключаться между экспериментами. Основная область странички делится на отображение видео (с права) и редактор кода (с лева).

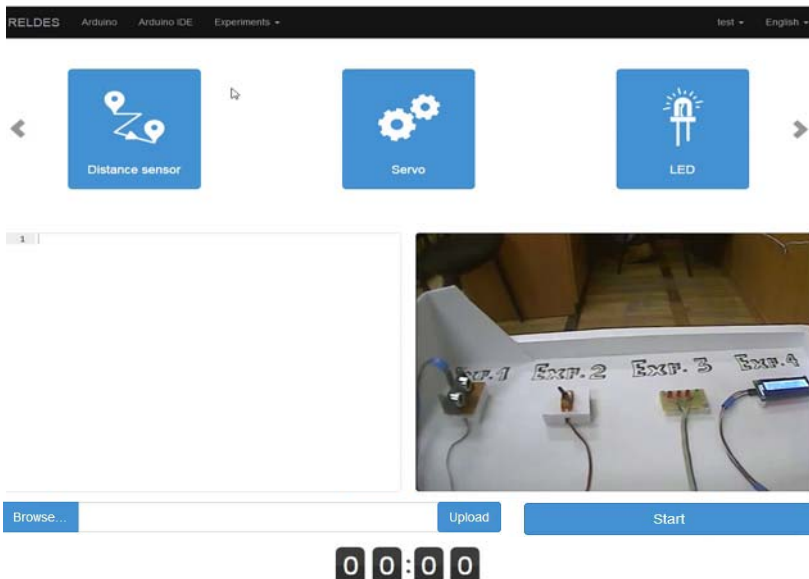


Рисунок 1.3 – Страница работы с экспериментом

1.1 Методика работы с удаленной лабораторией RELDES

Для начала работы нужно зарегистрироваться. Для этого на главной странице (рис.1.1) в главном меню кликаем на кнопку Log in. Появится окно авторизации (рис.1.4). Если пользователь впервые заходит на сайт, тогда нужно нажать на ссылку Registration. Появится регистрационное окно (рис.1.5), в котором вводим логин, пароль и подтверждение пароля. После нажимаем ОК.

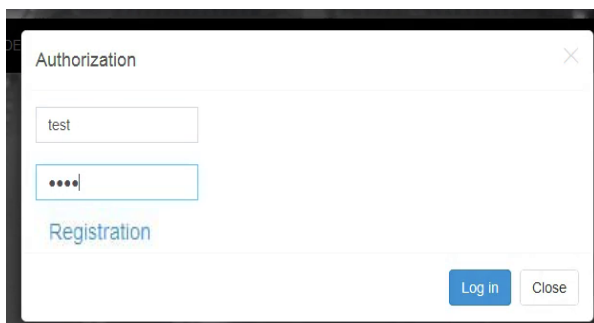


Рисунок 1.4 – Окно авторизации



Рисунок 1.5 – Регистрационное окно

Веб-сервис содержит справочную информацию о платах Arduino и среде разработки Arduino IDE, которая поможет новичку получить нужную информацию для начала программирования плат Arduino. На рис.1.6 изображено вид страницы, которая содержит информацию об Arduino платах. Для того, чтобы перейти на нее, в главном меню программы нужно выбрать пункт «Arduino».

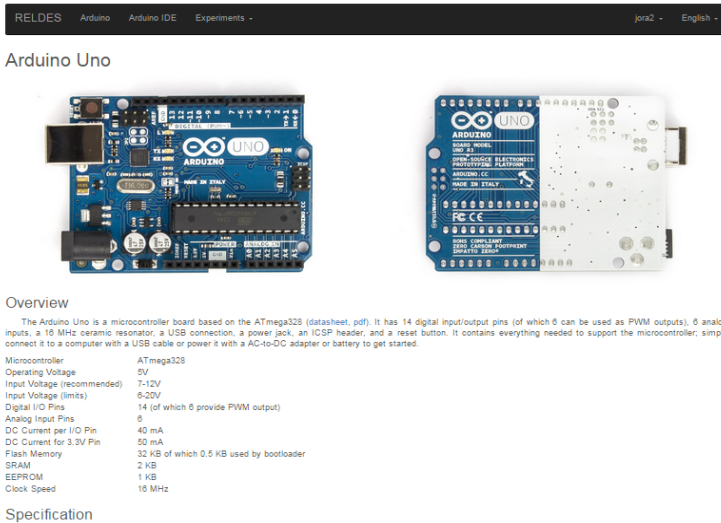


Рисунок 1.6 – Страница Arduino

В свою очередь, после выбора пункта меню «Arduino IDE», формируется страница Arduino IDE (рис.1.7).

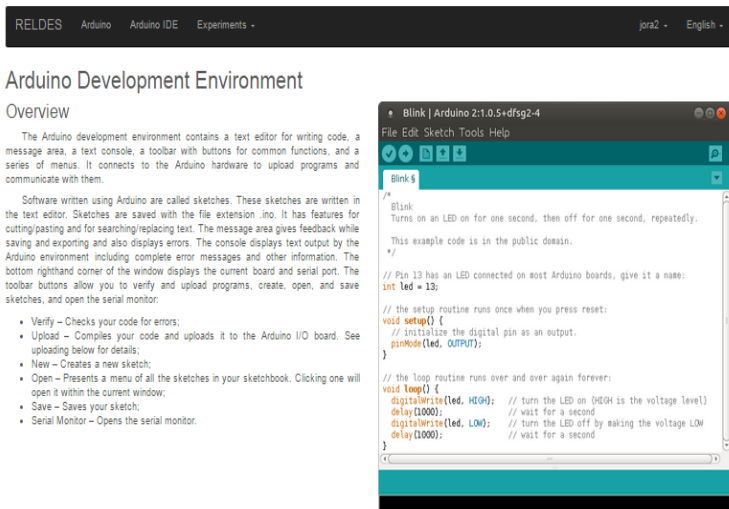


Рисунок 1.7 – Страница Arduino IDE

Далее, чтобы перейти на страницу эксперимента, нужно в главном меню «Experiment» (рис.1.2) выбрать пункт подменю с

названием эксперимента, с которым необходимо поработать. В данном примере выбираем подпункт меню «LED».

Для того чтобы начать эксперимент, кликаем кнопку Start. После этого, он может использовать пример кода приведенный на странице или написать собственный. Для подготовки и отправки кода на компиляцию существует два способа:

1. написать код в редакторе кода (рис.1.8) и отправить на компиляцию с помощью кнопки Upload;
2. или с помощью кнопки Browse загрузить файл со своим кодом программы. После загрузки файла нажать кнопку Upload.

Если в коде есть ошибка, то в рабочей области под редактором кода появится сообщение об этой ошибке. После завершения редактирования или если сервер не выдал сообщение об ошибках, надо отправить код на загрузку в микроконтроллер платы эксперимента. Для этого, нажав на кнопку Send, отправляем код на сервер для загрузки.

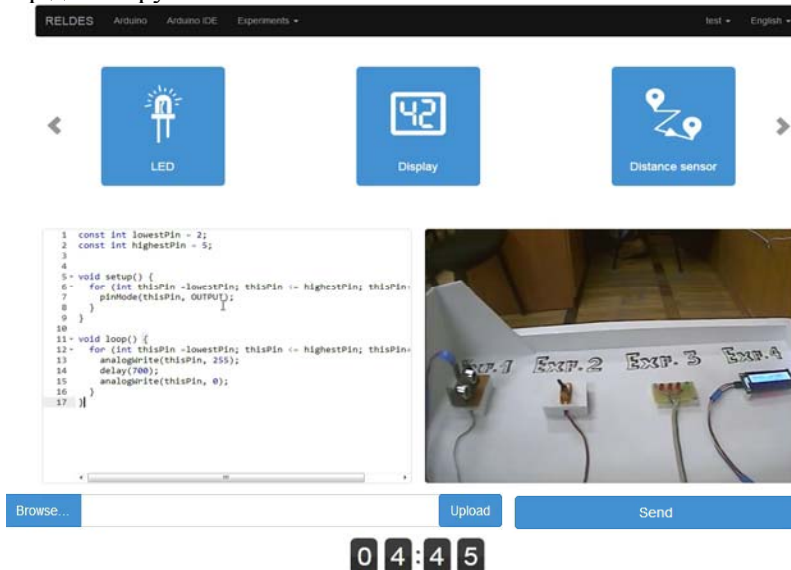


Рисунок 1.8 – Рабочая сессия эксперимента с кодом в редакторе

После этого, можно наблюдать результат работы программы с помощью веб-камеры в соответствующей области страницы.

2. Задания к самостоятельной работе

2.1 Эксперимент «Светофор»

В данном эксперименте производится управление набором светодиодов (рис.2.1). Управление осуществляется через цифровые выводы платы Arduino. Диоды подключены через токоограничивающие резисторы и имеют общее заземление. Принципиальная схема подключения светодиодов к плате изображена на рисунке 2.2.

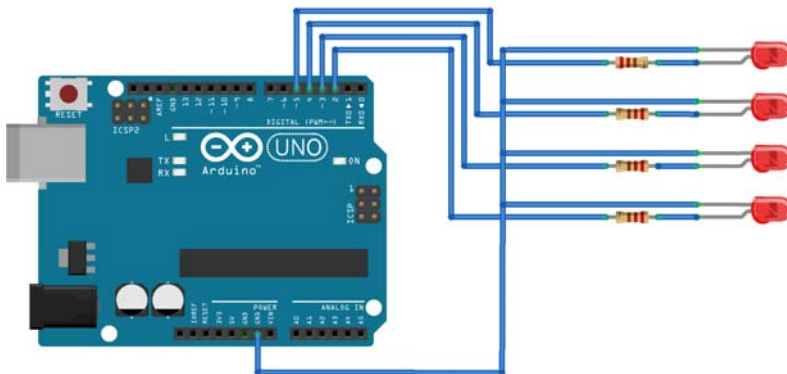


Рисунок 2.1 – Схематическое изображение

Программный код состоит из функций `setup` и `loop`. Первая выполняется единожды при включении платы. В данном случае в ней производится установка выводов с 2 по 5 в режим выхода с помощью вызова `pinMode`, параметрами которого являются номер вывода и его режим. Вторая функция `loop` выполняется в бесконечном цикле всё остальное время работы платы. В данном случае производится изменение яркости диодов, которые подключены к выводам 2-5. Для изменения яркости вызывается функция `analogWrite`, которая подаёт на вывод, указанный в своём первом параметре, напряжение, указанное во втором. Напряжение изменяется в диапазоне 0-5В, но функция принимает не значение вольтажа непосредственно, а число от 0 до 255, соответствующее напряжению на промежутке 0-5В. Изменение напряжение производится с помощью широтно-

импульсной модуляции. Так же в программе используется вызов `delay`, который приостанавливает работу платы на указанное кол-во времени (мс). Это нужно для того, чтобы работа выполнялась достаточно медленно, чтобы можно было заметить изменение яркости.

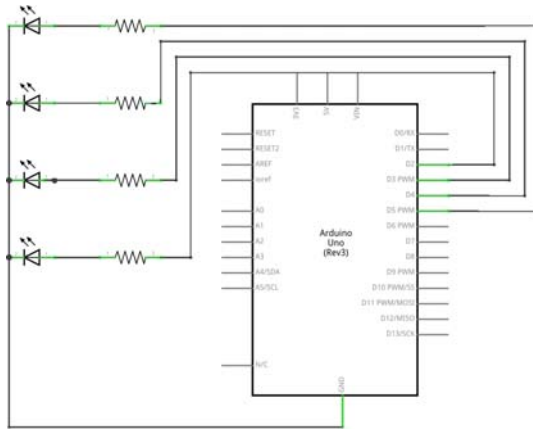


Рисунок 2.2 – Принципиальная схема эксперимента

Примеры программного кода:

Поочередное включение

```
const int lowestPin = 2;
const int highestPin = 5;

void setup() {
  for (int thisPin = lowestPin; thisPin <= highestPin;
thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}

void loop() {
  for (int thisPin = lowestPin; thisPin <= highestPin;
thisPin++) {
    analogWrite(thisPin, 255);
    delay(700);
    analogWrite(thisPin, 0);
  }
}
```

```
    }
}
```

В этом примере в цикле производится последовательное включение и выключения каждого диода с задержкой в 700мс.

Бегущая лента

```
const int lowestPin = 2;
const int highestPin = 5;
```

```
void setup() {
  for (int thisPin =lowestPin;  thisPin  <=  highestPin;
thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}
```

```
void loop() {
  for (int thisPin =lowestPin;  thisPin  <=  highestPin;
thisPin++) {
    analogWrite(thisPin, 255);
    delay(300);
  }
  for (int thisPin =lowestPin;  thisPin  <=  highestPin;
thisPin++) {
    analogWrite(thisPin, 0);
    delay(300);
  }
}
```

В этом примере в функции loop производится поочерёдное зажигание всех диодов в первом цикле и их последовательно выключении во втором. Эти циклы чередуются.

Плавное изменения яркости

```
const int lowestPin = 2;
const int highestPin = 5;
```

```
void setup() {
  for (int thisPin =lowestPin; thisPin <= highestPin; thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}
```

```
void loop() {
  for (int thisPin =lowestPin; thisPin <= highestPin; thisPin++) {
```

```

for (int brightness = 0; brightness < 255; brightness++) {
    analogWrite(thisPin, brightness);
    delay(2);
}

for (int brightness = 255; brightness >= 0; brightness--) {
    analogWrite(thisPin, brightness);
    delay(2);
}

delay(100);
}
}

```

Тут поочередно для каждого диода производится плавное изменение яркости от нуля до максимума и обратно. Изменение яркости производится во вложенных циклах.

2.2 Эксперимент «Серводвигатель»

В данном эксперименте производится управление серводвигателем. Управление производится последовательным кодом через цифровой вывод. Серводвигатель имеет три вывода: питание, земля и сигнальный.

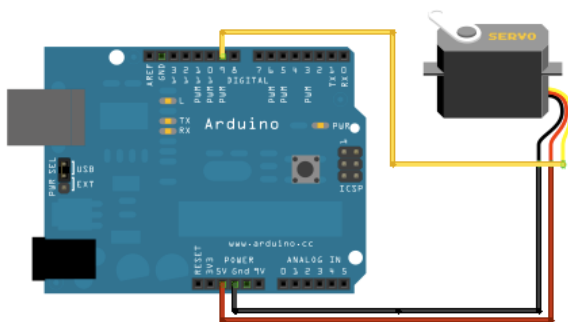


Рисунок 2.3 – Схематическое изображение

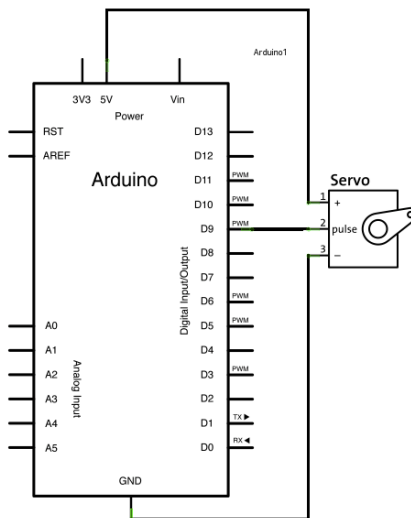


Рисунок 2.4 – Принципиальная схема

В первую очередь следует отметить, что в программе подключена библиотека `Servo.h`, отвечающая за управление серводвигателем. Она предоставляет возможность пользоваться классом `Servo`, через который и производится управление. Можно увидеть, как в начале программы создаётся объект этого класса.

Программный код состоит из функций `setup` и `loop`. Первая выполняется единожды при включении платы. В ней производится установка управляющего выхода для серводвигателя на вывод 9 с помощью вызова метода `attach` объекта `myservo`. Вторая функция `loop` выполняется в бесконечном цикле всё остальное время работы платы. В данной функции вызываются команды управления серводвигателем, задающие его позицию в градусах (`write`). Следует отметить, что диапазон работы серводвигателей может отличаться, в зависимости от двигателя и не всего равен 0-180. Команда `delay` используется для приостановки работы платы на указанное кол-во миллисекунд. Это необходимо потому, как скорость работы серводвигателя механически ограничена, и он не способен менять позицию на столько быстро, на сколько контроллер способен этого запрашивать.

Пример программного кода:

Поворот на две позиции

```

#include <Servo.h>
Servo myservo;
int pos = 0;

void setup()
{
  myservo.attach(9);
}

void loop()
{
  if(0 == pos) pos = 120;
  else pos = 0;
  myservo.write(pos);
  delay(2000);
}

```

Тут серводвигателю поочередно задаются 2 позиции. Между этим производится задержка в 2с, чтобы двигатель успел совершить поворот до получения следующей команды.

Плавное вращение

```

#include <Servo.h>

Servo myservo;

int pos = 0;

void setup()
{
  myservo.attach(9);
}

void loop()
{
  for(pos = 0; pos < 180; pos += 1)
  {
    myservo.write(pos);

    delay(15);
  }
  for(pos = 180; pos >= 1; pos -= 1)
  {
    myservo.write(pos);

```

```

    delay(15);
}
}

```

Тут производится плавный поворот двигателя. Для этого в цикле каждые 15мс вызывается функция смены позиции двигателя с разницей в 1 градус.

2.3 Эксперимент «Ультразвуковой датчик расстояния»

В данном эксперименте производится чтение показаний ультразвукового датчика расстояния. Датчик имеет три вывода: питание, земли и сигнальный. Сигнальный вывод подключен на цифровой вывод платы, данные поступают в последовательном коде.

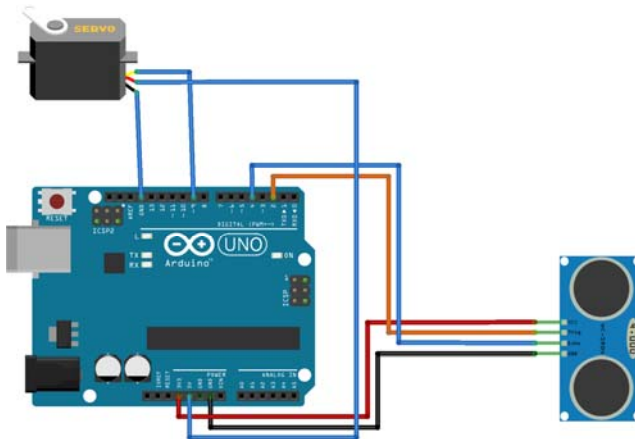


Рисунок 2.5 – Схематическое изображение

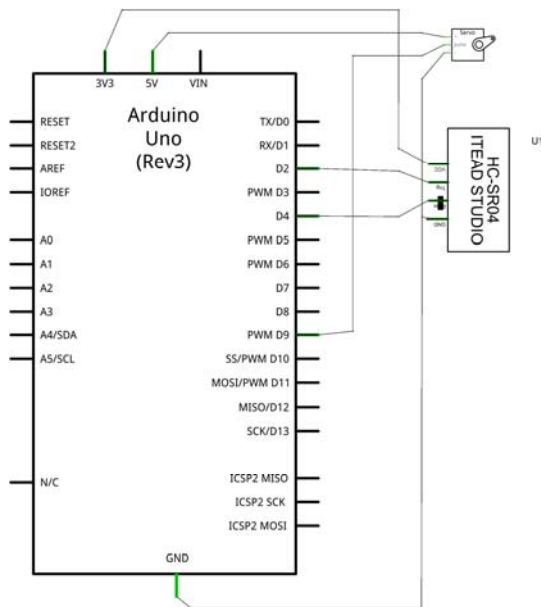


Рисунок 2.6 – Принципиальная схема

В данном примере передача показателей датчика расстояния на компьютер производится через последовательный интерфейс. Для работы с ним используется класс `Serial`, через который и происходит всё взаимодействие с интерфейсом.

Программный код состоит из функций `setup` и `loop`. Первая выполняется единожды при включении платы. В данном случае в ней производится инициализация последовательного интерфейса с частотой с помощью вызова `begin`, параметром которого служит частота передачи данных, также производится установка режимов для выводов датчика функцией `pinMode`, параметрами которой служат номер вывода и режим. Также перед описанием функции `setup` объявляются переменные, хранящие номера выводов, к которым будет подключен датчик расстояния. Вторая функция `loop` выполняется в бесконечном цикле всё остальное время работы платы.

Работа с датчиком расстояния производится в два постоянно чередующихся этапа: запрос на измерение и чтение показателей. Так как датчик является ультразвуковым, показатели, возвращаемые им,

представляют собой время, за которое ультразвуковой сигнал прошёл до объекта и обратно. Введя этого, после чтения показаний следует преобразовать значения, чтобы вычислить расстояние.

Для запроса на датчик в течении более 10 мкс подаётся логическая единица. Перед и после неё передаётся логический ноль. Это нужно для того, чтобы удостовериться, что датчик правильно считал команду. Данные операции производятся с помощью вызовов `digitalWrite`, параметрами которого являются номер выхода и значение (`LOW` или `HIGH`). Между вызовами производится задержка с помощью функции `delayMicroseconds`.

Для чтения показателей датчика считывается длительность входящего сигнала. Это производится с помощью вызова функции `pulseIn`, параметрами которой являются номер входа и сигнал, длительность которого следует узнать (логический 0 или 1). Эта функция возвращает длительность сигнала в мкс. Полученное значение заносится в переменную `duration`.

После получения этого значения, с помощью функций `microsecondsToCentimeters` и `microsecondsToInches` производится арифметическое преобразование значения времени в расстояние в см и дюймах соответственно.

По заверению этих операций производится пересылка полученной информации через серийный интерфейс (`Serial`) с помощью вызова `print`.

Пример программного кода:

Измерение дистанции

```
const int trigPin = 2;
const int sigPin = 4;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(sigPin, INPUT);
}

void loop()
{
  long duration, inches, cm;

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
```

```

digitalWrite(trigPin, HIGH);
delayMicroseconds(5);
digitalWrite(trigPin, LOW);

duration = pulseIn(sigPin, HIGH);

inches = microsecondsToInches(duration);
cm = microsecondsToCentimeters(duration);

Serial.print(inches);
Serial.print("in, ");
Serial.print(cm);
Serial.print("cm");
Serial.println();

delay(100);
}

long microsecondsToInches(long microseconds)
{
    return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds)
{
    return microseconds / 29 / 2;
}

```

Датчик расстояния с серводвигателем

```

#include <Servo.h>

Servo myservo;

int positions[] = { 0, 30, 60, 90, 120, 150, 180 };
int pos_num = 7;
int pos = 0;

const int trigPin = 2;
const int echoPin = 4;

void setup()
{
    Serial.begin(9600);
    myservo.attach(9);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
}

void loop()

```

```

{
  myservo.write( positions[pos] );
  pos++;
  if( pos == pos_num ) pos = 0;

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  Serial.print( pulseIn(echoPin, HIGH) / 29 / 2 );
  Serial.print("cm");
  Serial.println();

  delay(500);
}

```

В данном эксперименте датчик расстояния поворачивается с помощью серводвигателя. Это позволяет измерять дистанцию до разных объектов. На стенде на разном удалении от датчика расположены перегородки. Пользователю предлагается проводить повороты двигателя с последующими снятиями показаний датчика расстояния.

Положение двигателя задаётся вызовом write, параметром которого является угол поворота. В данном случае значения углов хранятся в массиве и поочерёдно задаются двигателю. Команда на измерение расстояния подается датчику при помощи импульса длительностью 10мкс. Для этого последовательно производится запись значений (0 и 1) с помощью вызова DigitalWrite в соответствующий вывод. После измерения расстояния производится чтение показаний датчика (время прохода ультразвукового сигнала) с помощью вызова pulseIn. Вывод данных пользователю производится через последовательный интерфейс с помощью функции print.

2.4 Эксперимент «ЖК дисплей»

В данном эксперименте производится управление ЖК дисплеем. Управление изображением дисплея производится через цифровые выводы по протоколу последовательной передачи данных I2C.

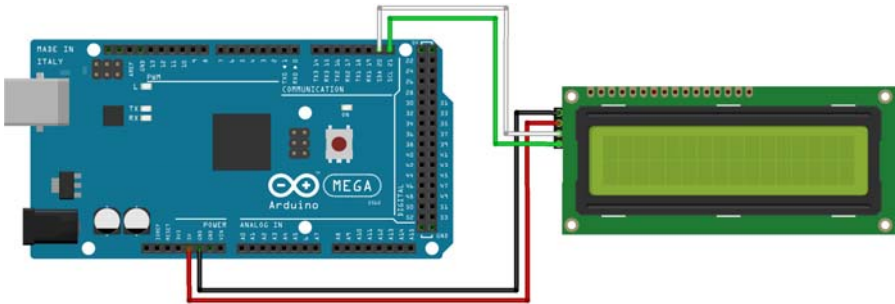


Рисунок 2.7 – Схематическое изображение

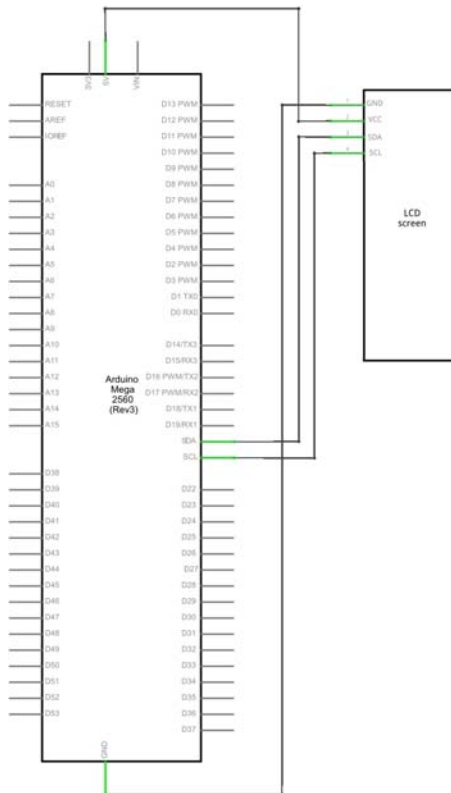


Рисунок 2.8 – Принципиальная схема

В первую очередь следует отметить, что в данной программе подключены библиотеки `LiquidCrystal_I2C.h` (отвечающая за управление ЖК-дисплеем) и `Wire.h` (реализующая I2C протокол). Библиотеки предоставляют возможность пользоваться классом `LiquidCrystal_I2C`, через который и производится управление. Можно увидеть, как в начале программы создаётся объект этого класса. В данном случае параметры соответствуют встроенному в экран контроллеру и изменять их не рекомендуется.

Программный код состоит из функций `setup` и `loop`. Первая выполняется единожды при включении платы. В данном случае в ней производится установка параметров работы ЖК-дисплея с помощью вызова `begin` с параметрами, означающими количество столбцов и рядков символов. Для включения/выключения встроенной подсветки дисплея используются функции `backlight/noBacklight`. Вторая функция `loop` выполняется в бесконечном цикле всё остальное время работы платы. В данном случае в ней производится установка курсора в начало второй строки (нумерация производится с нуля) с помощью вызова `setCursor` и печать времени работы платы в секундах. Функция `print` вызывается для вывода на экран указанного набора символов. Вызов `clear` позволяет очистить содержимое экрана, а функция `delay` приостанавливает работу платы на указанное в миллисекундах время.

Пример программного кода

Hello, world!

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

void setup(){
    lcd.begin(16,2);
    lcd.backlight();
}

void loop(){
    lcd.setCursor(0,0);
    lcd.print("Hello, world!");
    delay(1000);
    lcd.setCursor(0,1);
```

```

    lcd.print("I am display");
    delay(1000);
    lcd.clear();
    delay(1000);
}

```

В данном примере в первую строку экрана выводится сообщение «Hello, world!», а после секундной задержки во вторую строку выводится «I am display». Спустя секунду после этого экран очищается. Действия повторяются циклично.

Бегущая строка

```

#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
int col_offset = 0;
int row_offset = 0;

void setup(){
    lcd.begin(16,2);
    lcd.backlight();
}

void loop(){
    lcd.setCursor(col_offset, row_offset);
    lcd.print("Running text");
    delay(100);
    lcd.clear();
    col_offset = (col_offset + 1) % 16;
    if(15 == col_offset) row_offset = (row_offset + 1) % 2;
}

```

В данном примере реализована бегущая строка. В каждой итерации цикла производится сдвиг начальной позиции курсора, с которой на следующей итерации будет производиться вывод текста.

Самостоятельная работа № 2

Реализация эксперимента «Плата быстрого прототипирования» в среде удаленной лаборатории GOLDI

1. Теоритические сведения

1.1 Rapid Prototyping Board (Плата быстрого прототипирования) лаборатории GOLDI

Плата быстрого прототипирования (RP) показана на рисунке 2.1.

Режимы работы:

1) Онлайн *быстрое прототипирование* цифровых систем:

- проектирование (текстовое, PCB, графы конечных автоматов);
- программирование CPLD посредством интерфейса удаленной лаборатории;
- тестирование проекта через RIA и онлайнową RP плату.

2) Онлайн *тестирование* цифровых систем:

- выявлять функция проекта (черный ящик);
- находить неисправности в работе хорошо известного проекта;
- путем манипулирования входами или загрузкой таблицы истинности или тестовых векторов;
- CPLD не программируемые студентами.

Основные компоненты Rapid Prototyping Board: MAX V – 5M1270Z (CPLD от Altera); индикаторы входных сигналов (8 переключателей, 8 кнопок, 2 поворотные шестигранные энкодеры); светодиодные выходы (4 7-сегментных дисплеев, 1 светодиодный дисплей бар); другие компоненты (10 МГц кварцевый генератор, частотный синтезатор, пьезоэлектрический генератор, инкрементный энкодер, UART (через USB), 25-контактный разъем SUB-D).

Схема взаимодействия RP платы с инфраструктурой удаленной лаборатории показана на рисунке 2.2:

- связь с удаленной инфраструктурой лаборатории через "Interconnection" FPGA;
- все входы платы удаляются из печатной платы;
- заменен на прямое подключение к выходам "Interconnection" FPGA;
- устанавливает входные сигналы в соответствии входу пользователя через пользовательский интерфейс;

- читает выходы непосредственно;
- "Interconnection" ПЛИС подключается к BPU в инфраструктуре удаленной лаборатории [16].

Веб-интерфейса пользователя реализован как HTML5 RIA (Rich Internet Application) показан на рис. 2.3.

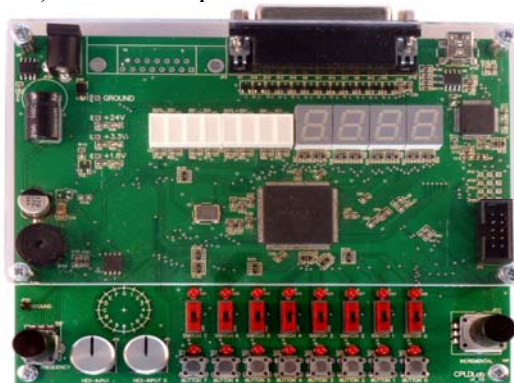


Рисунок 2.1 – Плата быстрого прототипирования

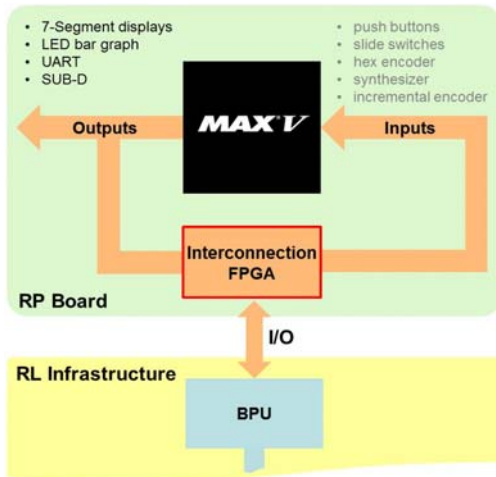


Рисунок 2.2 - Схема взаимодействия RP платы с инфраструктурой удаленной лаборатории

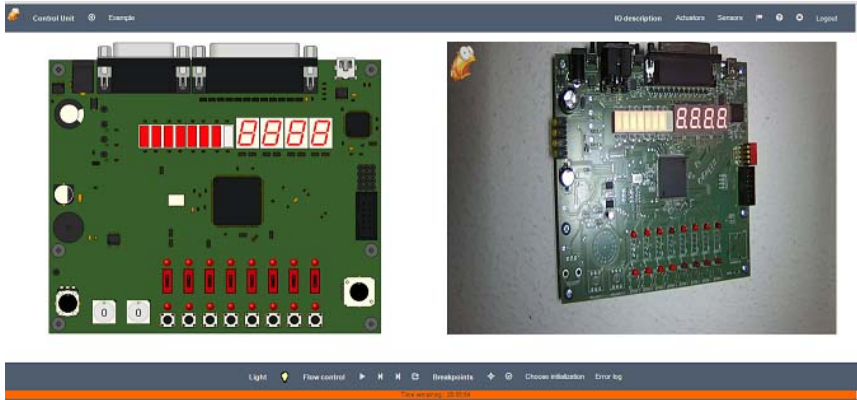


Рисунок 2.3 Web-интерфейс пользователя - RIA

Основные компоненты интерфейса:

- загрузка синтезированного CPLD кода проекта для автоматического программирования CPLD в удаленной лаборатории;
- визуальная модель платы RP, чтобы управлять всеми входами виртуально и наблюдать выходные сигналы платы RP;
- наблюдение через веб-камеры в удаленной лаборатории, чтобы смотреть результаты действия пользователя напрямую.

Этапы разработки для Цифровых Систем показаны на Рисунке 2.4 [16]. Схема показывает основные блоки:

- общие средства разработки: реализация задач управления, например, с использованием IDE Альтера; студенты могут использовать "самодельное" ядро IP библиотек;
- загрузка синтезированного проекта на плату RP;
- программирование автономной платы RP;
- онлайн программирование платы RP посредством интерфейса удаленной лаборатории.

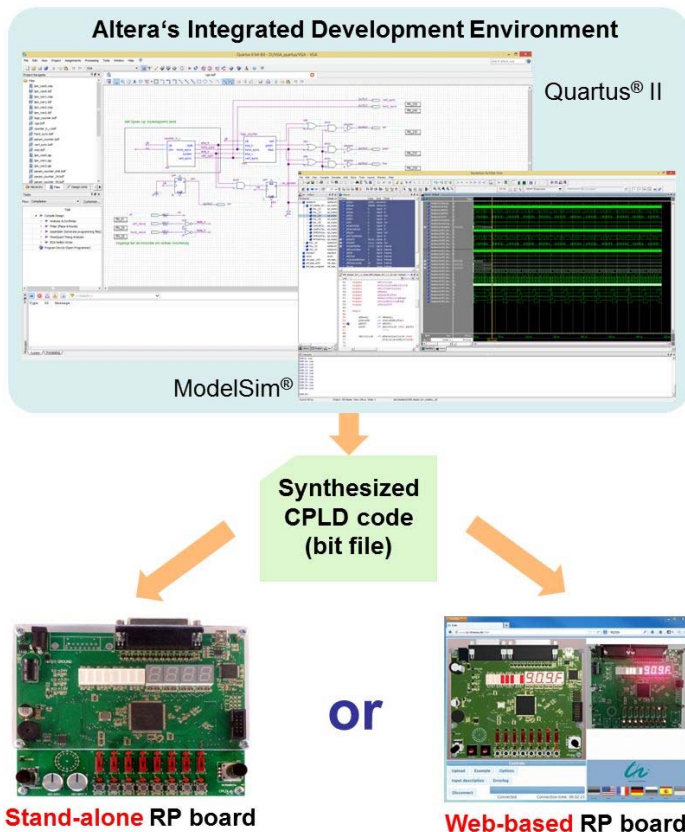


Рисунок 2.4 – Этапы разработки для Цифровых Систем

Метод проектирования на основе ввода текста дизайна с помощью логических уравнений, таблиц истинности или языков описания аппаратных средств (AHDL, VHDL или Verilog) (рис. 2.5).

Графический метод проектирования на основе ввода конструкцию с помощью блок-схемы и/или схематические диаграммы (рис. 2.6).

Встроенный редактор FSM где вводится дизайн с помощью автоматных графов (FSM) (рис. 2.7).

Имитация дизайна с использованием различных инструментов моделирования (например, Qsim или ModelSim) (рис.2.8) [16].

Рисунок 2.6 Графический метод проектирования

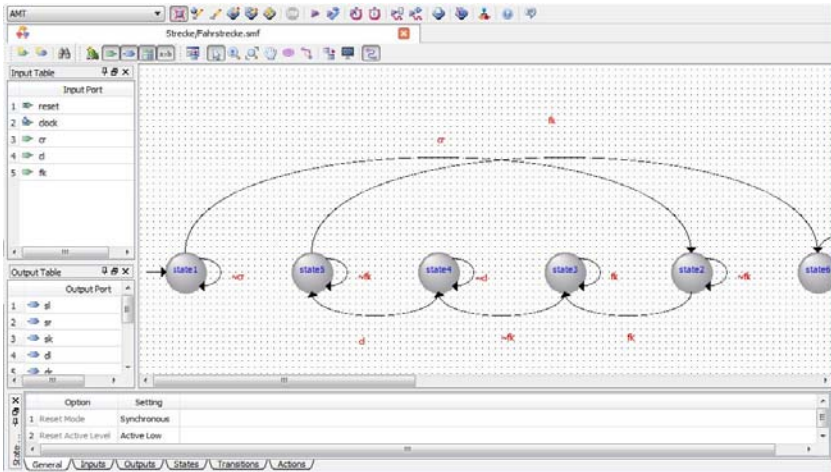


Рисунок 2.7 – FSM метод проектирования

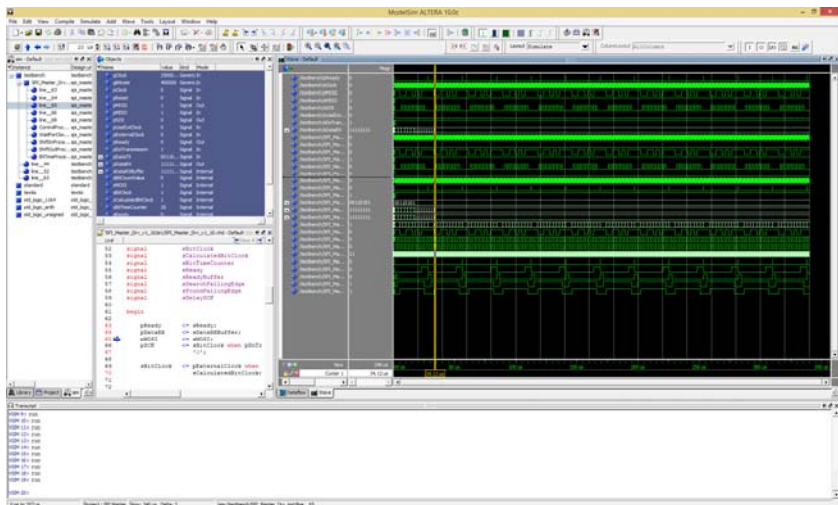


Рисунок 2.8 — Имитация дизайна

2. Задание к самостоятельной работе

GOLDI эксперимент – Работа с 7-сегментными дисплеями

Далее, мы рассмотрим на примере работу 7-сегментного дисплея платы быстрого прототипирования с использованием среды Quartus II от Altera.

MAX V CPLD (Complex Programmable Logic Device) плата RP имеет четыре 7-сегментных дисплея. Применение низкого логического уровня на сегмент заставляет его светиться, и применяя высокий логический уровень выключает его. Каждый сегмент на дисплее обозначается индексом от 0 до 7, с позициями приведенными на рис.2.9. Таблица 2.1 показывает назначение выводов CPLD на первый 7-сегментный дисплей.

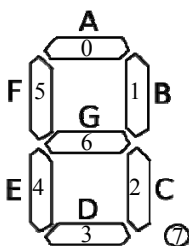


Рисунок 2.9 — Позиция и индекс каждого сегмента

С помощью этого дисплея мы можем отобразить цифры от 0 до 9. Для отображения 1, используя 7 сегментный дисплей, мы должны сделать выход В и С с высоким логическим уровнем. В двоичном коде 1 представлена как '0001'. Таким образом, '0001' необходимо преобразовать в '1001111', используя BCD 7-сегментный декодер. Таблица 2.2 представляет таблицу истинности BCD 7 сегментного декодера.

В качестве бинарных входов мы будем использовать 4 тумблера платы. Плата предоставляет 8 тумблеров (ползунки). Эти переключатели предназначены для использования в качестве уровня чувствительных входных данных в схему. Каждый переключатель подключается непосредственно к выводу на плате быстрого прототипирования. Когда переключатель находится в положении ВНИЗ это обеспечивает низкий логический уровень (0 вольт) на CPLD, а когда переключатель находится в положении ВВЕРХ - обеспечивает высокий логический уровень. Список имен контактов на

плате быстрого прототипирования, которые подключены к тумблерам приведены в таблице 2.3.

Запустите Quartus II. Выберите **File->New Project Wizard...** и "Next". Теперь необходимо назвать свой проект и рабочий каталог. Назовем проект "7_Seg_Decoder". Нажмите два раза кнопку "Next" (рис. 2.10).

Таблица 2.1 – Назначение выводов для первого 7-сегментного дисплея

Signal Name	CPLD pin number	Description
Hex0[7]	PIN_84	Output
Hex0[6]	PIN_88	Output
Hex0[5]	PIN_89	Output
Hex0[4]	PIN_91	Output
Hex0[3]	PIN_93	Output
Hex0[2]	PIN_85	Output
Hex0[1]	PIN_86	Output
Hex0[0]	PIN_87	Output

Таблица 2.2 - Таблица истинности

Binary Inputs				Decoder Outputs							Outputs
X3	X2	X1	X0	a	b	c	d	e	f	g	
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	1	0	0	1	1	1	1	1
0	0	1	0	0	0	1	0	0	1	0	2
0	0	1	1	0	0	0	0	1	1	0	3
0	1	0	0	1	0	0	1	1	0	0	4
0	1	0	1	0	1	0	0	1	0	0	5
0	1	1	0	0	1	0	0	0	0	0	6
0	1	1	1	0	0	0	1	1	1	1	7
1	0	0	0	0	0	0	0	0	0	0	8
1	0	0	1	0	0	0	0	1	0	0	9
1	0	1	0	0	0	0	1	0	0	0	A
1	0	1	1	1	1	0	0	0	0	0	b
1	1	0	0	0	1	1	0	0	0	1	C
1	1	0	1	1	0	0	0	0	1	0	d
1	1	1	0	0	1	1	0	0	0	0	E
1	1	1	1	0	1	1	1	0	0	0	F

Таблица 2.3 - Назначение контактов для тумблеров

Signal Name	CPLD pin number	Description
Switch[7]	PIN_55	Input
Switch[6]	PIN_58	Input
Switch[5]	PIN_60	Input
Switch[4]	PIN_62	Input
Switch[3]	PIN_66	Input
Switch[2]	PIN_68	Input
Switch[1]	PIN_70	Input
Switch[0]	PIN_72	Input

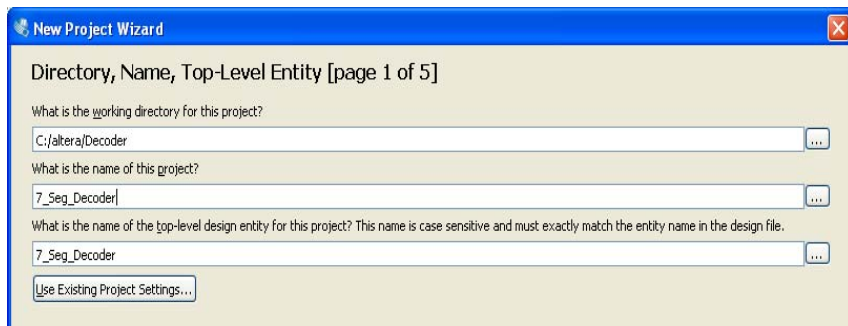


Рисунок 2.10 Окно для ввода имени проекта

В следующем окне (рис.2.11) выбираем CPLD, которую необходимо. В этом случае выбираем MAX V 5M1270ZT144C5. Устройство также может быть изменено позже (**Assignments->Device**). После этого выбираем "Next".

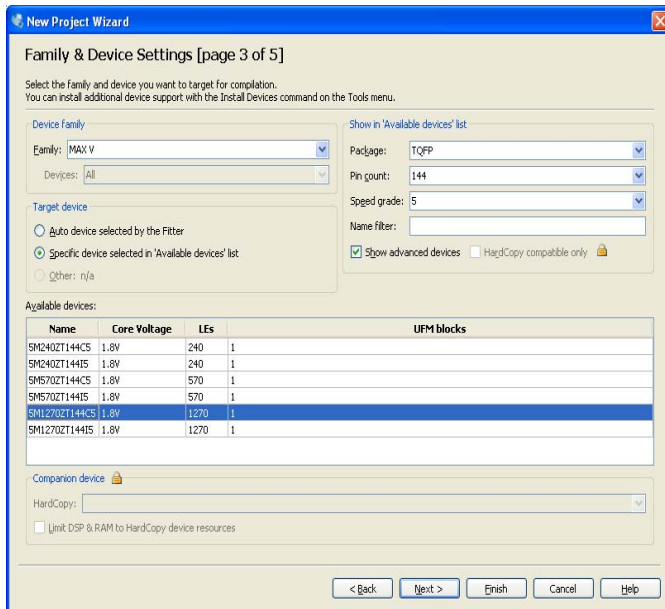


Рисунок 2.11 — Окно для выбора CPLD

В следующем окне (рис. 2.12) выбираем ModelSim-Altera, как инструмент симулирования EDA. После выбираем "Next" и "Finish". Для создания новых графических файлов выберите **File->New**. В списке выбираем графические файлы "Block Diagram/Schematic File" (рис. 2.13).

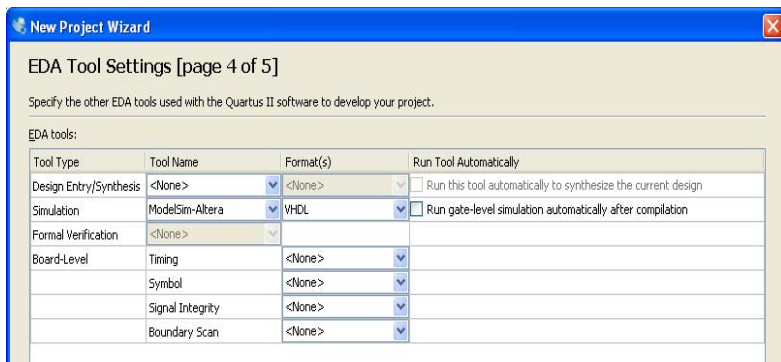


Рисунок 2.12 — Окно для выбора EDA инструментов

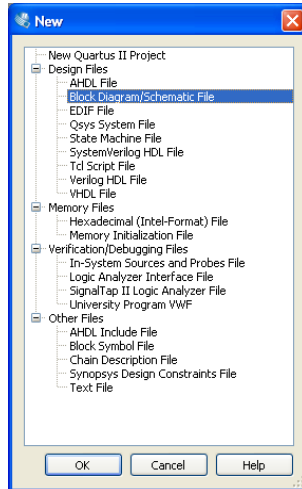


Рисунок 2.13 — Окно для создания новых проектных файлов

Сначала сохраните файл под именем "SevSegmentDecoder".

Следующий этап включает в себя рисование комбинационной логической схемы для каждого выходного сигнала. Входы обозначаем как x_3 , x_2 , x_1 и x_0 , а выходы a , b , ..., g для 7 сегментов (рис.2.14).

a

$\begin{array}{c ccc} & x_0 & x_1 & x_2 & x_3 \\ \hline x_3 & & & & \end{array}$	0	1	1	0	
	0	0	1	1	
0	0	0	1	0	0
0	1	1	0	0	0
1	1	0	1	0	0
1	0	0	0	1	0

b

$\begin{array}{c ccc} & x_0 & x_1 & x_2 & x_3 \\ \hline x_3 & & & & \end{array}$	0	1	1	0	
	0	0	1	1	
0	0	0	0	0	0
0	1	0	1	0	1
1	1	1	0	1	1
1	0	0	0	1	0

c

$\begin{array}{c ccc} & x_0 & x_1 & x_2 & x_3 \\ \hline x_3 & & & & \end{array}$	0	1	1	0	
	0	0	1	1	
0	0	0	0	0	1
0	1	0	0	0	0
1	1	1	0	1	1
1	0	0	0	0	0

d

$\begin{array}{c ccc} & x_0 & x_1 & x_2 & x_3 \\ \hline x_3 & & & & \end{array}$	0	1	1	0	
	0	0	1	1	
0	0	0	1	0	0
0	1	1	0	1	0
1	1	0	0	1	0
1	0	0	0	0	1

e

$\begin{array}{c ccc} & x_0 & x_1 & x_2 & x_3 \\ \hline x_3 & & & & \end{array}$	0	1	1	0	
	0	0	1	1	
0	0	0	1	1	0
0	1	1	1	1	0
1	1	0	0	0	0
1	0	0	1	0	0

f

$\begin{array}{c ccc} & x_0 & x_1 & x_2 & x_3 \\ \hline x_3 & & & & \end{array}$	0	1	1	0	
	0	0	1	1	
0	0	0	1	1	1
0	1	0	0	1	0
1	1	0	1	0	0
1	0	0	0	0	0

g

$\begin{array}{c ccc} & x_0 & x_1 & x_2 & x_3 \\ \hline x_3 & & & & \end{array}$	0	1	1	0	
	0	0	1	1	
0	0	1	1	0	0
0	1	0	0	1	0
1	1	1	0	0	0
1	0	0	0	0	0

$$a = \overline{x_1}x_0(\overline{x_3 \oplus x_2}) + \overline{x_3}x_2\overline{x_1}x_0 + x_3\overline{x_2}x_1x_0;$$

$$b = x_2\overline{x_1}(x_3 \oplus x_0) + x_3x_1x_0 + x_2x_1\overline{x_0};$$

$$c = \overline{x_3}x_2\overline{x_1}x_0 + x_3x_2(x_1 + \overline{x_0});$$

$$d = \overline{x_3}x_2\overline{x_1}x_0 + \overline{x_3}x_2\overline{x_1}x_0 + x_2x_1x_0 + x_3\overline{x_2}x_1\overline{x_0};$$

$$e = \overline{x_3}x_2\overline{x_1} + \overline{x_3}x_0 + \overline{x_2}x_1x_0;$$

$$f = x_3x_2\overline{x_1}x_0 + \overline{x_3}x_2(x_0 + x_1) + \overline{x_3}x_1x_0;$$

$$g = \overline{x_3}x_2\overline{x_1} + \overline{x_3}x_2\overline{x_1}x_0 + \overline{x_3}x_2x_1x_0.$$

Рисунок 2.14 — Выходные значения для логической схемы 7-сегментного декодера

Из приведенной выше таблицы истинности (таблица 2.3), мы получаем упрощенную k-карту для общего катодного 7-сегментного

декодера с целью разработки комбинационной схемы. И после упрощения, мы получаем выходные значения для создания логической схемы 7-сегментного декодера.

Для создания логической схемы BCD для 7-сегментного декодера на панели инструментов находится значок для вставки символов, Symbol Tool (с правой стороны от A - символ) Здесь можно выбрать NOT, OR и AND символы. (Библиотеки: primitives->logic) (рис.2.15).

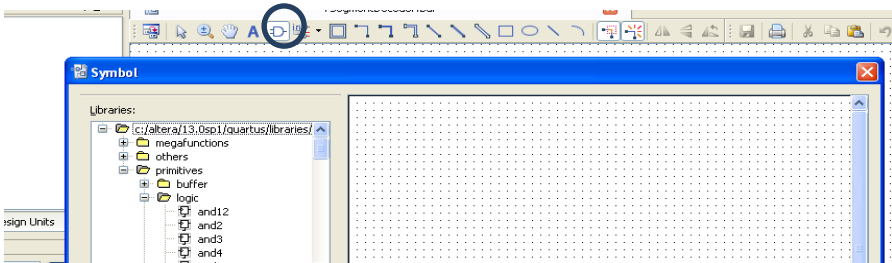


Рисунок 2.15 — Окно Symbol Tool

Рядом с Symbol Tool есть значок Pin Tool - установка входных пинов и семь выходных пинов. Нажмите дважды на имя контакта, чтобы изменить его. Соедините элементы, как показано на рис. 2.16. Сохраните файл еще раз.

Установите дизайн как дизайн верхнего уровня: **Project-> Set как Top-Level Entity**. Компиляция дизайн **Processing->Start Compilation**.

Для того, чтобы использовать 7-сегментный декодер для создания основной конструкции, мы создадим символьный файл для 7-сегментного декодера. С помощью этого файла мы сможем позже вставить полную логику 7-сегментного декодера, в другую блок-схемы, одним символом. Для этого выберите **File ->Create/Update ->Create Symbol Files for Current File**.

Для имитации дизайна должен быть создан файл моделирования: **File->New->University Program VWF**. Сохранить файл. Файл моделирования должен иметь такое же имя, как схематичный файла — "SevSegmentDecoder". Добавьте входы и выходы в файле моделирования дизайна: **Edit->Insert->Insert Node or Bus....** Установить значения для каждого входа.

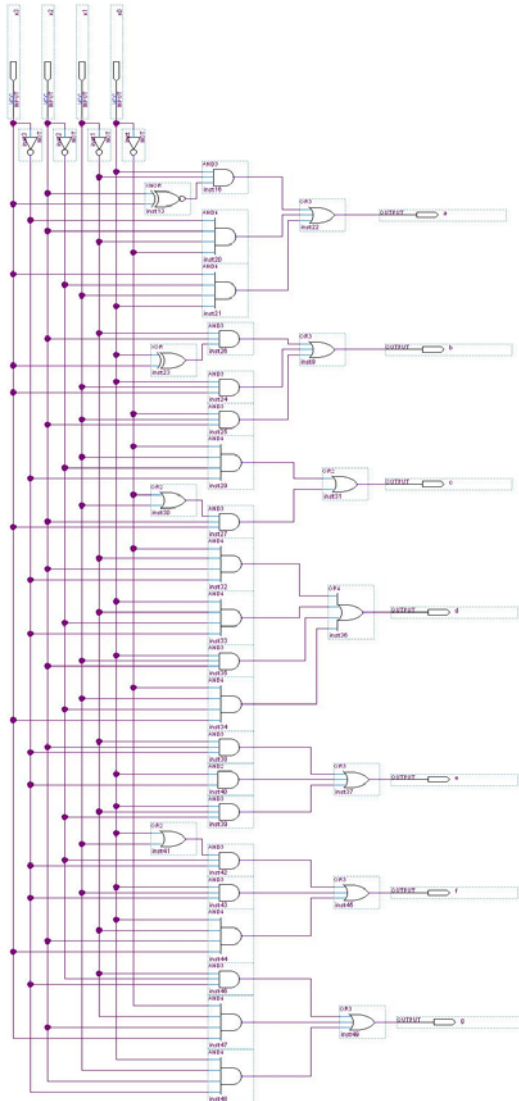


Рисунок 2.16 — Логическая схема проекта_1

Для того кликните "Node Finder...". Далее "List" all Nodes и выбираем все (рис.2.17). Кликните дважды "ОК".

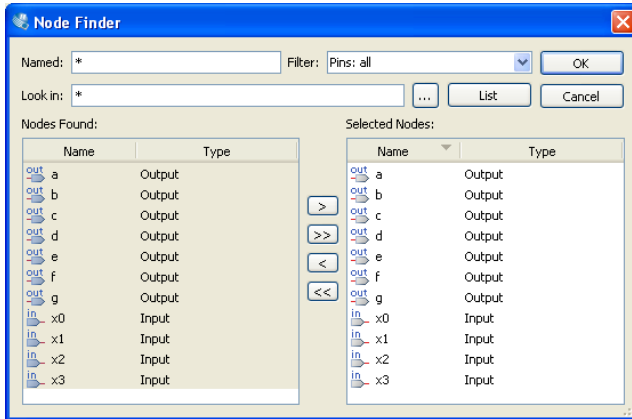


Рисунок 2.17 - Окно Node Finder.

В этом файле необходимо определить поведение входов. Для всех сигналов легче использовать "Count Value" в панели инструментов. Чтобы создать группу, выберите все входы с помощью правой кнопки мыши и "Ctrl". Затем выберите **Edit->Grouping->Group...** Установите значения других входов, как поведение, которое вы хотите проверить. (можно не изменять выходные значения.) Например, как на рисунке 2.18.

Вы можете начать моделирование **Simulation->Run Functional Simulation**. Проверьте результаты моделирования (рис.2.19).

Создать новый схематический файл "Seg_Decoder_test.bdf" (рис.2.20).

Установить в качестве высокоуровневой сущности. Начать компиляцию. После того, для Распиновки дизайна необходимо использовать Pin Planner: **Assignments->Pin Planner** (рис.2.21.). Номера пинов используйте из таблицы 2.1 и таблицы 2.3. После того, как распиновка дизайна сделана, схема выглядит как показано на рисунке 2.22.

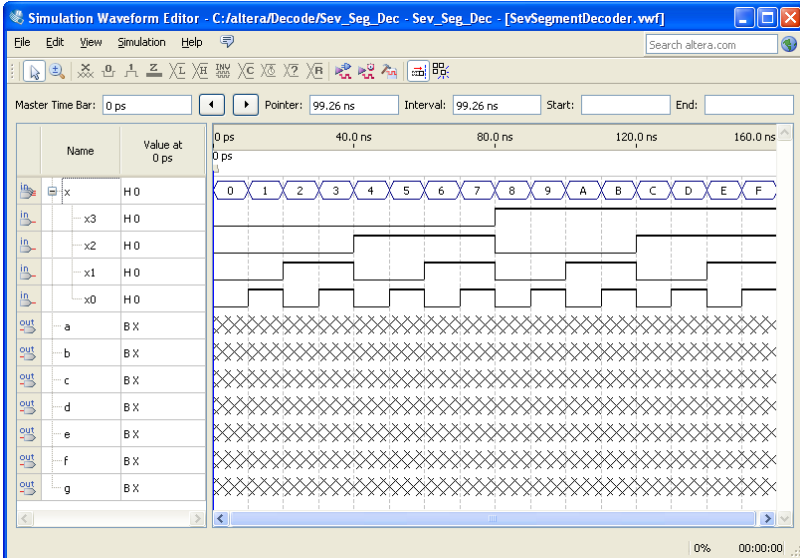


Рисунок 2.18 — Окно для определения входных поведений и создания групп входов

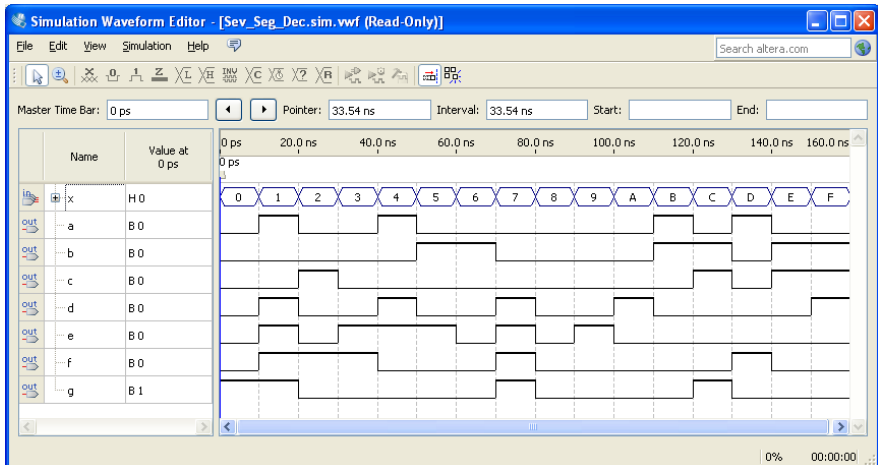


Рисунок 2.19 — Результаты моделирования

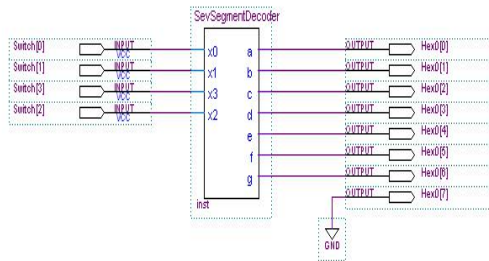


Рисунок 2.20 — Логическая схема проекта _2

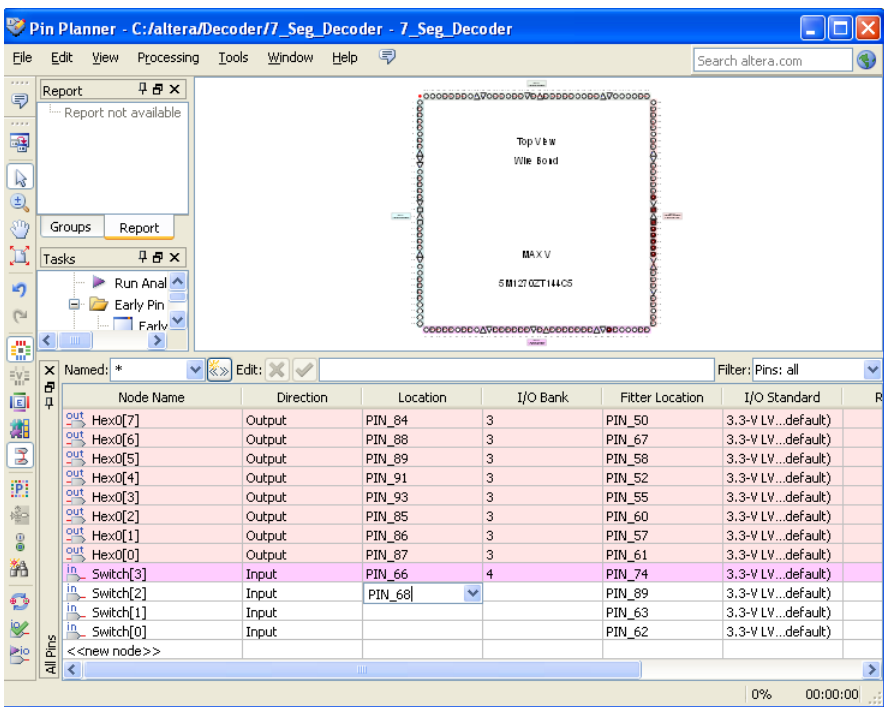


Рисунок 2.21 — Окно Pin Planner

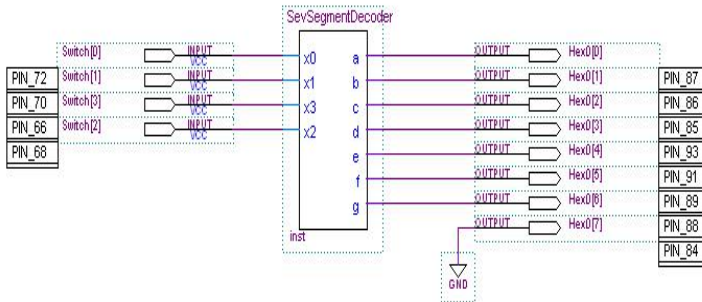


Рисунок 2.22 — Логическая схема проекта _2 после распиновки

Сохраните проект. Откомпилируйте снова. В директории проекта в папке "output_files" вы можете найти файл "7_Seg_Decoder.pof". Этот файл можно использовать для программирования CPLD платы. После того, как результат был запрограммирован на плату RP лаборатории Goldi (автономное или через веб-интерфейс), можно проверить работу дизайна (рис. 2.23, 2.24).

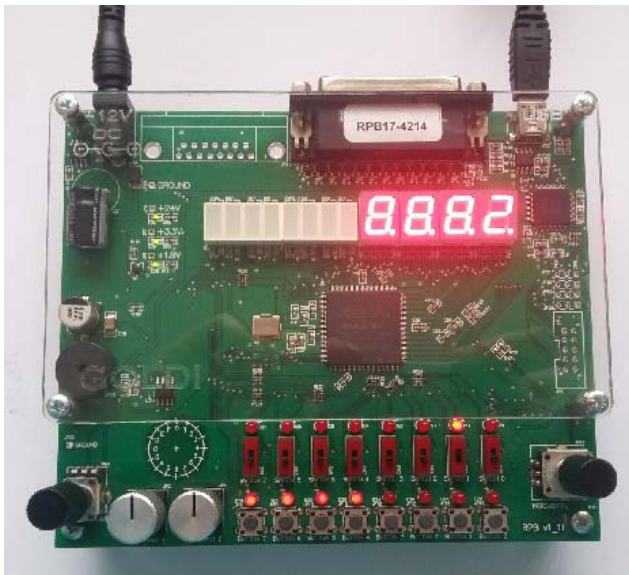


Рисунок 2.23 — Тестирование дизайна на автономной плате быстрого прототипирования (Stand alone RP board) GOLDI

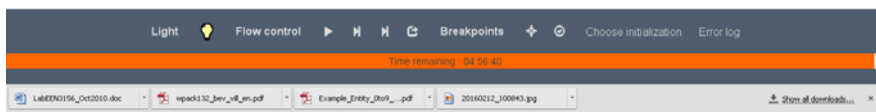
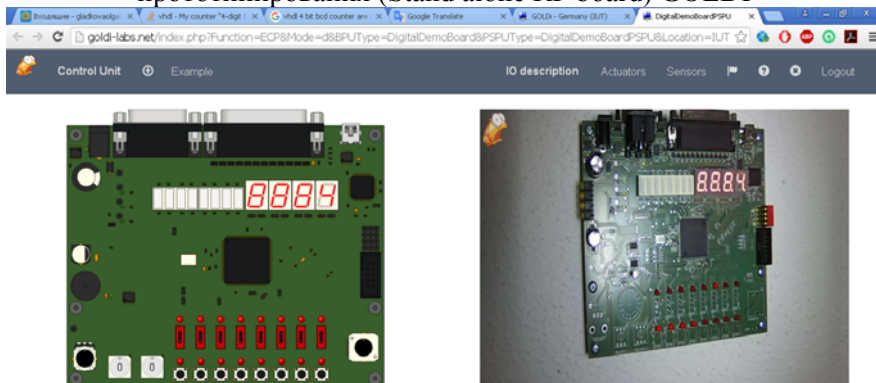


Рисунок 2.24 — Тестирование дизайна на онлайн плате быстрого прототипирования GOLDI

Другой способ создать 7-сегментный декодер используя аппаратные языки программирования:

- VHDL – Very high speed integrated circuits **H**ardware **D**escription **L**anguage;
- AHDL - Altera **H**ardware **D**escription **L**anguages;
- Verilog HDL.

Здесь мы рассмотрим только два примера языка: VHDL и AHDL. Создайте новый файл: **File->New->VHDL File**. Ниже приведена программа для BCD декодера 7-сегментного декодера. Скопируйте код в VHDL файл.

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY seg_vhdl IS
    PORT (
        x : IN  STD_LOGIC_VECTOR(3 DOWNTO 0);--BCD input
        y : OUT  STD_LOGIC_VECTOR(6 DOWNTO 0);--7bit
        decoded output
    );

```

```

END seg_VHDL;
ARCHITECTURE TrueTable OF seg_VHDL IS
BEGIN
    WITH x SELECT
        y<="0000001" WHEN "0000", --0
        "1001111" WHEN "0001", --1
        "0010010" WHEN "0010", --2
        "0000110" WHEN "0011", --3
        "1001100" WHEN "0100", --4
        "0100100" WHEN "0101", --5
        "0100000" WHEN "0110", --6
        "0001111" WHEN "0111", --7
        "0000000" WHEN "1000", --8
        "0000100" WHEN "1001", --9
        "0001000" WHEN "1010", --A
        "1100000" WHEN "1011", --b
        "0110001" WHEN "1100", --C
        "1000010" WHEN "1101", --d
        "0110000" WHEN "1110", --E
        "0111000" WHEN "1111", --F
        "1111111" WHEN others;

```

```
END TrueTable;
```

Сохраните файл как “seg_vhdl”, установите как файл верхнего уровня, скомпилируйте и создайте символьный файл.

Для создания проекта AHDL выберите: **File->New->AHDL File**. Скопируйте код в файл.

```

SUBDESIGN  seg_ahdl
(
    p[3..0]      : INPUT;
    a,b,c,d,e,f,g : OUTPUT;
)
BEGIN
    TABLE
p[3..0] => a,b,c,d,e,f,g;
    H"0" => 0, 0, 0, 0, 0, 0, 1;

```

```

H"1" => 1, 0, 0, 1, 1, 1, 1;
H"2" => 0, 0, 1, 0, 0, 1, 0;
H"3" => 0, 0, 0, 0, 1, 1, 0;
H"4" => 1, 0, 0, 1, 1, 0, 0;
H"5" => 0, 1, 0, 0, 1, 0, 0;
H"6" => 0, 1, 0, 0, 0, 0, 0;
H"7" => 0, 0, 0, 1, 1, 1, 1;
H"8" => 0, 0, 0, 0, 0, 0, 0;
H"9" => 0, 0, 0, 0, 1, 0, 0;
H"A" => 0, 0, 0, 1, 0, 0, 0;
H"B" => 1, 1, 0, 0, 0, 0, 0;
H"C" => 0, 1, 1, 0, 0, 0, 1;
H"D" => 1, 0, 0, 0, 0, 0, 1;
H"E" => 0, 1, 1, 0, 0, 0, 0;
H"F" => 0, 1, 1, 1, 0, 0, 0;
      END TABLE;
END;
```

Сохраните файл как “seg_ahdl”, скомпилируйте и после успешной компиляции создайте символьный файл, теперь можно добавить этот символ в файл проекта.

Контрольные вопросы

1. До какого вывода подключен встроенный светодиод на платформе Arduino Uno?
2. Какие операторы среды Arduino надо использовать, чтобы создать проект управления светодиодами?
3. Что такое RX и TX на плате Arduino?
4. Сколько всего входов/выходов имеет платформа Arduino Uno? Перечислите основные из них
5. Что такое сервопривод?
6. Как происходит подключение и управление сервоприводом?
7. Основные функции Arduino для работы с дисплеем?
8. Какие пины на плате Arduino используются для создания проекта работы с дисплеем?
9. Что такое LCD I2C модуль?
10. Какие основные компоненты Платы быстрого прототипирования?
11. Какие типы плат быстрого прототипирования существуют?
12. Какими методами можно создать дизайн?
13. Как производится распиновка в Quartus и зачем ее делать?
14. Сколько сегментов у 7 сегментного дисплея платы RP?
15. Нарисуйте позиции и индексы каждого сегмента в 7-сегментном дисплее.
16. Напишите таблицу ASCII чисел от 0 до F (используя 10 и 16 системы счисления)
17. Что такое таблица истинности?
18. Какие типы простых логических элементов вы знаете?
19. Как работает символ “not4” в проекте декодера?
20. Как работает декодер?
21. Что вы знаете об AHDL и VHDL? В чем разница?

ЛИТЕРАТУРА

1. Віддалений та віртуальний інструментарій в інжинірингу: монографія за аг. ред.. проф.. Карстена Хенке / А.В. Пархоменко, Г.В. Табунщик, М.О. Поляков, О.М. Гладкова, Т.І. Каплієнко, Т.Ю. Ларіонова. – Запоріжжя: Дике поле, 2015. – 250 с. ISBN 978-966-2752-74-8
2. Henke, K. Fields of Applications for Hybrid Online Labs / Karsten Henke, Steffen Ostendorff, Heinz-Dietrich Wuttke, Tobias Vietzke, Christian Lutze // International Journal of Online Engineering (iJOE), Vol 9 (2013) – Access mode: <http://online-journals.org/i-joe/article/view/2542>
3. Henke, K. Using Interactive Hybrid Online Labs for Rapid Prototyping of Digital Systems / K. Henke, G. Tabunshchyk, H.D. Wuttke, T. Vietzke, St. Ostendorff // Remote Engineering & Virtual Instrumentation - REV2014 , Porto, Portugal, February 2014, pp.61-66
4. Henke, K. A Grid Concept for Reliable, Flexible and Robust Remote Engineering Laboratories/ Karsten Henke, Steffen Ostendorff, Heinz- Dietrich Wuttke, Stefan Vogel // Remote Engineering & Virtual Instrumentation - REV2012 , Bilbao, Spain, July 2012
5. Henke K. Web-based Rapid Prototyping of Digital Systems/ Karsten Henke, Silvia Krug // International Conference on Interactive Computer-Aided Blended Learning, Antigua, Guatemala, November 2011, pp. 22-27
6. Wuttke H- D. Remote and Virtual Laboratories in Problem-Based Learning Scenarios / Heinz- Dietrich Wuttke, Raimund Ubar, Karsten Henke//, IEEE International Symposium on Multimedia, Taichung, Taiwan, December 2010, pp.377-382