Министерство образования и науки Российской Федерации

Государственное образовательное учреждение высшего профессионального образования

«Вологодский Государственный Технический Университет»

Кафедра: Автоматики и вычислительной техники

Дисциплина: Электронные устройства систем управления

Курсовой проект

«Разработка кодового замка на Arduino

на базе микроконтроллера ATMega168»

Выполнил: Веселов Д.А.

Группа: ЭПО-31

Проверил: Сердюков Н.А.

Вологда,

2011

Содержание

[Техническое задание 3](#_Toc314106951)

[Введение 4](#_Toc314106952)

[1. Аналитический обзор 5](#_Toc314106953)

[2. Проектирование электронного замка 9](#_Toc314106954)

[3. Расчёт и выбор основных компонентов 10](#_Toc314106955)

[3.1. Актуатор (привод) центрального замка 10](#_Toc314106956)

[3.2. Полевой транзистор IRF520 10](#_Toc314106957)

[3.3. Arduino на базе микроконтроллера ATMega168 11](#_Toc314106958)

[3.4. LCD-экран WH1602B-YYK-CTK 11](#_Toc314106959)

[3.5. Тактовые кнопки TC-0404 11](#_Toc314106960)

[4. Исходный код программы для работы устройства 13](#_Toc314106961)

[5. Разработка конструкции устройства 17](#_Toc314106962)

[Заключение 18](#_Toc314106963)

[Литература 19](#_Toc314106964)

Техническое задание

Разработать принципиальную схему, печатную плату и программу прошивки для кодового дверного замка на плате Arduino на базе микроконтроллера ATMega168. Кодовый замок должен выполнять такие функции как:

1. При нажатии на цифровые кнопки на дисплей выводится код с клавиатуры;
2. Несколько кнопок для цифрового ввода кода, одна кнопка чтобы стереть предыдущее значение;
3. После ввода правильного пароля на дисплей выводится фраза «Дверь открыта» и дверь открывается;
4. После 8 секунд простоя без нажатия кнопок все введённые значения стираются с дисплея;
5. Невозможно изменить пароль без «перепрошивки» микроконтроллера.

Так же кодовый замок должен быть спроектирован из доступных элементов и легко воспроизводим начинающими радиолюбителями.

Введение

В настоящее время радиолюбительский рынок заполнен устройствами, которые используются в системах оповещения и безопасности. Эти устройства, от самых простых и до сложных, собраны, как правило, на микросхемах триггеров, но большинство решений предлагаемых на радиотехнических сайтах или не работают, или не имеют печатных плат.

Разработанное устройство доступно для повторения начинающим радиолюбителям-конструкторам, не обладающим глубокими теоритическими знаниями в электронике, и может быть использовано для охраны таких объектов, как квартира, офис или дача от несанкционированного доступа.

Кодовый замок – очень удобная и практичная вещь. Не нужно постоянно носить с собой связку металлических ключей в кармане, чтобы открыть дверь – для этого достаточно просто вспомнить код.

Кодовые замки можно разделить на несколько групп, но самыми популярными остаются только две: механические и электронные. В этой курсовой работе спроектирован электронный замок.

1. Аналитический обзор

Сейчас существует множество [1] решений для сборки кодовых замков, но не все они просты для повторения начинающим или они имеют недостатки, которые можно выяснить только после того, как устройство будет собрано. К тому же, в интернете есть и полностью нерабочие схемы, после сборки которых пропадает желание пробовать ещё.

Было рассмотрено несколько вариантов устройств на различных микросхемах и с различными подходами для ввода кода.

Самый простой вариант основан на 4017 микросхеме [2] и состоит всего из 2 транзисторов, 5 диодов, 5 резисторов, 4 конденсаторах и 9 кнопок. Он спроектирован так, что используются лишь 4 кнопки S6-S9, остальные (S1-S5) приводят к сбрасыванию регистра на 15 выводе, что приводит к обнулению вводимых до этого символов.

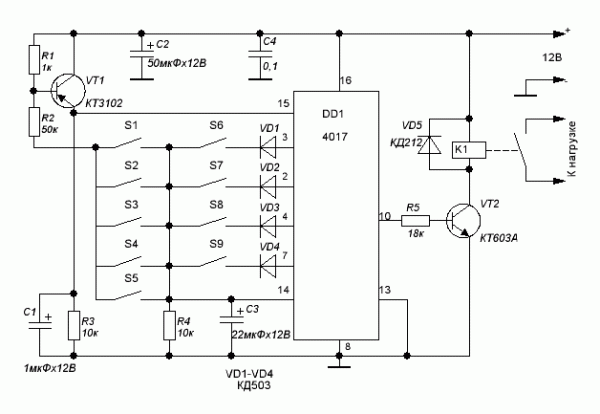


Рисунок 1.1. Принципиальная схема кодового замка на 4017 микросхеме

Но данная схема оказалась не работоспособной, так как диод VD5 поставлен не той стороной, транзистор VT1 выбран не подходящий, поэтому если подключать какой-то другой, то нужно менять сопротивления R1 и R2.

Вывод по данному варианту: схема простая, но нужно дорабатывать её, чтобы устройство работало как было задумано, к тому же отсутствует чертёж печатной платы.

Второй вариант [3] электронного замка основан на открытии двери с помощью ключей – электронных таблеток iButton. Это устройство более интересно с той точки зрения, что оно основано на программируемом микроконтроллере AT89C2051 фирмы Atmel.

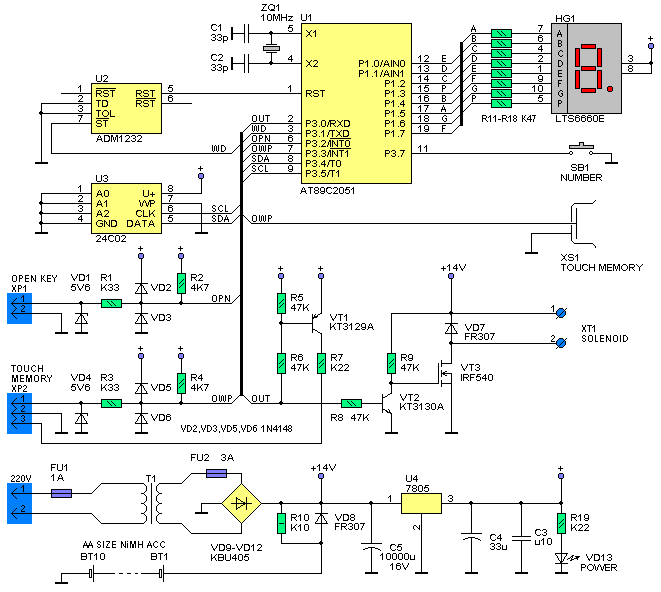


Рисунок 1.2. Принципиальная схема кодового замка на AT89C2051 микроконтроллере

Данное устройство запрограммировано на 9 ключей-таблеток, при этом у каждого ключа свой уникальный код. Любой ключ можно заменить на другой, перейдя в режим программирования (он «вшит» в микроконтроллер).

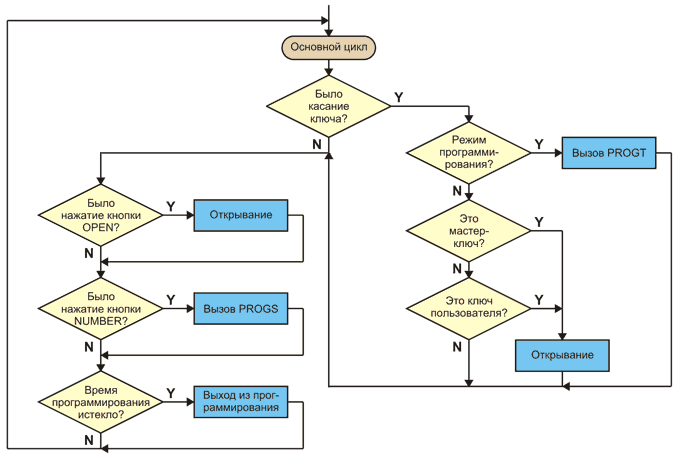


Рисунок 1.3. Алгоритм работы электронного замка

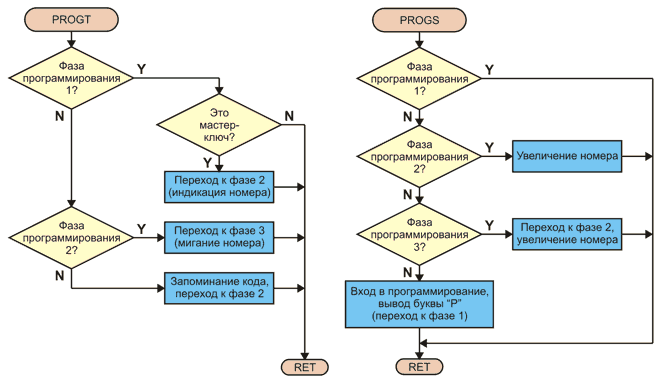


Рисунок 1.4. Подпрограммы, используемые при программировании кода ключа

Выводы по данному варианту: не хватает чертежа печатной платы, схема сложна к воспроизведению и «прошивка» написана на Assembler’е – сложно переделать её под свои нужны.

1. Проектирование электронного замка

Чтобы разработать какое-то устройство на базе Arduino нужно вначале спроектировать его в системе симулирования. В Multisim 10.1 в стандартной поставке отсутствуют микроконтроллеры, в Proteus ISIS 7 есть нужный микроконтроллер ATMega168, но нет купленного экрана (LCD-дисплея), поэтому я остановился на ещё молодой, но перспективной разработке студентов из немецкого университета прикладных наук в Потсдаме – программе для прототипирования **Fritzing**. В этой программе очень наглядный и удобный интерфейс, позволяющий собирать элементы на макетной плате так же, как это делается и в жизни. В ней была разработана первая и вторая версия принципиальной схемы.

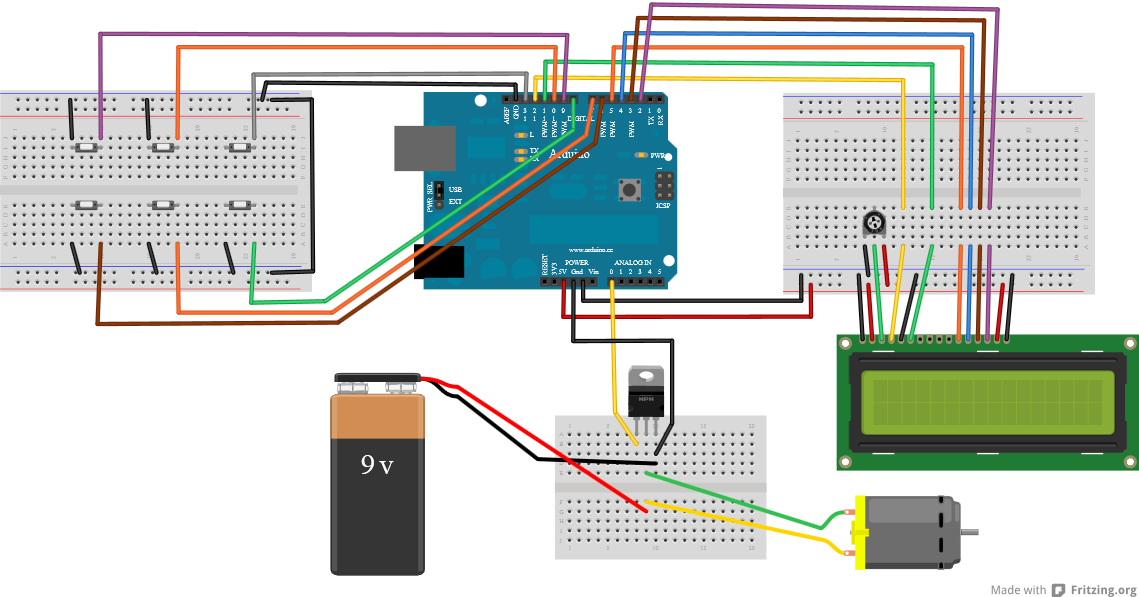


Рисунок 2.1. Макетная плата проектируемого электронного замка, выполненная в программе Fritzing.

Схему следует рассматривать с учётом того, что здесь за место мотора будет машинный актуатор центрального замка и источник питания для него будет 12В.

1. Расчёт и выбор основных компонентов
   1. Актуатор (привод) центрального замка

Выбран типовой 2-х проводной привод, работающий от напряжения 12В и при токе 2-3А. Выбор основан тем, что он проще в управлении, нежели 5-проводной, у него меньший ток потребления, ход выходного штока больше, а так же он дешевле и доступнее в автомагазинах.

* 1. Полевой транзистор IRF520

Выбран полевой транзистор в корпусе TO-220, так как он позволяет работать с гораздо большими мощностями, а управление затвором осуществляется исключительно при помощи напряжения: ток через затвор, в отличии от биполярных транзисторов, не идёт. Нужно кратковременно включить привод, запитываемый от большого тока, поэтому следует использовать транзистор. Так Arduino, при выдаваемых на контакт 5В выдерживает ток в 40мА, а привод работает при 2-3А. При подключении таких нагрузок напрямую чип выйдет из строя. Кроме того для работоспособности требуется напряжение большее, чем 5В, а Ардуино с выходного контакта больше 5 В не может выдать в принципе.

При подключении биполярного транзистора следует использовать токоограничивающий резистор между контактом Arduino и базой транзистора, чтобы при подаче управляющего напряжения не образовалось короткое замыкание по маршруту микроконтроллер — транзистор — земля.  
Расчёт токоограничивающего резистора производится по формуле:

|  |  |
| --- | --- |
| Graph | (3.2.1) |

Здесь *U* – напряжение выдаваемое на выходном контакте Arduino, обычно используют *U*=5В, *Ud* – это падение напряжения на самом транзисторе. Оно зависит от материала, из которого он изготовлен и обычно составляет 0.3-0.6В.

А если вместо биполярного транзистора использовать полевой, то можно обойтись без токоограничительного резистора – это связано с тем, что затвор в таких транзисторах управляется исключительно напряжением: ток на участке микроконтроллер — затвор — исток отсутствует [4]. А благодаря своим высоким характеристикам схема с использованием MOSFET, позволяет управлять очень мощными компонентами.

* 1. Arduino на базе микроконтроллера ATMega168

Arduino выбран потому, что для него не нужен прошивальщик и все программы пишутся на языке Wiring, которого на самом деле не существует, как и не существует и компилятора Wiring – написанные на нём программы преобразуются (с минимальными изменениям) в программы на языке C/C++, и затем компилируются компилятором AVR-GCC. Так что, фактически, используется специализированный для микроконтроллеров AVR вариант C/C++ [5].

Выбрана плата с микроконтроллером ATMega168 из-за того, что она содержит в себе 16 Кбайт памяти для загрузки программ (у ATMega8 их 8Кбайт) и ОЗУ у неё 1 Кбайт.

* 1. LCD-экран WH1602B-YYK-CTK

Выбран из-за того, что он дешевле других дисплеев, хорошо отображает русские символы и для него можно создавать свои собственные символы [6].

К дисплею подключается потенциометр номиналом 10 кОм, он нужен для регулировки контрастности.

* 1. Тактовые кнопки TC-0404

Выбранные кнопки хорошо подходят, чтобы нанести поверх них небольшую клавиатуру, и они имеют недорогую стоимость. В первой версии устройства использовались внешние подтягивающие резисторы (они нужны, чтобы не оставить вход в «подвешенном» состоянии [7]) на 150 Ом, но они загромождали схему печатной платы. Поэтому было решено использовать внутренние подтягивающие резисторы на 20-50кОм [8].

Так же было решено использовать 5 кнопок для ввода 5-значного кода и одну кнопку, чтобы стереть предыдущий символ. Количество кнопок обосновано тем, что если делать полноценную клавиатуру из 10-12 клавиш, то нужно будет использовать более сложную принципиальную схему с диодами и резисторами [9].

1. Исходный код программы для работы устройства

|  |
| --- |
| 1. // Библиотеки для работы с дисплеем 2. #include <LiquidCrystal.h> 3. #include <LiquidCrystalRus.h> 4. // Цифровые PINы - кнопки 5. const int button1 **=** 10**;** 6. const int button2 **=** 9**;** 7. const int button3 **=** 8**;** 8. const int button4 **=** 7**;** 9. const int button5 **=** 6**;** 10. const int button6 **=** 0**;** 11. // Цифровые PINы - дисплей 12. const int lcdRS **=** 12**;** 13. const int lcdE **=** 11**;** 14. const int lcdDB4 **=** 5**;** 15. const int lcdDB5 **=** 4**;** 16. const int lcdDB6 **=** 3**;** 17. const int lcdDB7 **=** 2**;** 18. // Инициализация драйвера дисплея 19. LiquidCrystalRus lcd**(**lcdRS**,** lcdE**,** lcdDB4**,** lcdDB5**,** lcdDB6**,** lcdDB7**);** 20. // Аналоговый PIN - управление тразистором 21. const int transistor **=** A0**;** 22. // Секретный код для открытия двери (транзистора) 23. char password**[**6**]** **=** "32145"**;** 24. // Буффер для ввода кода и счётчик введённых символов 25. char inputpassword**[**6**]** **=** "00000"**;** 26. int counter **=** 0**;** 27. // Нажатая кнопка 28. char pressedbtn **=** '0'**;** 29. // Время после нажатия на кнопку 30. int pressedbtnsecs **=** 0**;** 31. // Установка значений при включении 32. void setup**()** 33. **{** 34. // Устанавливаем кнопки для ввода 35. pinMode**(**button1**,** INPUT**);** 36. pinMode**(**button2**,** INPUT**);** 37. pinMode**(**button3**,** INPUT**);** 38. pinMode**(**button4**,** INPUT**);** 39. pinMode**(**button5**,** INPUT**);** 40. pinMode**(**button6**,** INPUT**);** 42. // Включение внутренних подтягивающих резисторов 43. digitalWrite**(**button1**,** HIGH**);** 44. digitalWrite**(**button2**,** HIGH**);** 45. digitalWrite**(**button3**,** HIGH**);** 46. digitalWrite**(**button4**,** HIGH**);** 47. digitalWrite**(**button5**,** HIGH**);** 48. digitalWrite**(**button6**,** HIGH**);** 49. // Инициализация дисплея 50. lcd**.**begin**(**16**,** 2**);** 51. // Вывод фразы на первой строке 52. lcd**.**print**(**"Введите код:"**);** 53. // Устанавливаем ввод на 2 строку дисплея 54. lcd**.**setCursor**(**0**,** 1**);** 56. // Обнулить вводимый пароль 57. pwdtonull**();** 59. // Чтобы по затвору-стоку не проходил ток при включении 60. analogWrite**(**transistor**,** 0**);** 61. **}** 62. // Бесконечный цикл 63. void loop**()** 64. **{** 65. // если больше 8 секунд назад была нажата кнопка 66. **if** **((**pressedbtnsecs **>** 1**)** **&&** **((**seconds**()** **-** pressedbtnsecs**)** **>** 8**))** 67. **{** 68. // стираем введённые значения и обнуляем счётчик 69. pwdtonull**();** 70. writeLcdPassword**();** 71. pressedbtnsecs **=** 0**;** 72. **}** 73. // Проверяем пароль 74. **if** **(**checkPassword**())** 75. **{** 76. // выводим текст на дисплей 77. lcd**.**setCursor**(**0**,**1**);** 78. lcd**.**print**(**"Дверь открыта!"**);** 79. // открываем транзистор 80. analogWrite**(**transistor**,** 1023**);** 81. delay**(**500**);** 82. analogWrite**(**transistor**,** 0**);** 84. // потом зануляем пароль 85. pwdtonull**();** 87. // сохраняем сколько секунд прошло после запуска Arduino 88. pressedbtnsecs **=** seconds**();** 89. **}** 91. // Смотрим какая кнопка нажата 92. pressedbtn **=** whatButton**();** 93. **if** **(**pressedbtn **>** '0'**)** 94. **{** 96. // Чтобы не выйти за границу массива пароля 97. **if** **(**counter **>=** 5**)** 98. **{** 99. counter **=** 0**;** 101. // и заодно стираем все символы, чтоб на дисплее не оставались старые значения 102. **for(**int i**=**0**;** i **<** 5**;** i**++)** 103. **{** 104. inputpassword**[**i**]** **=** '0'**;** 105. **}** 106. **}** 108. // сохраняем в массив введённых символов нажатую кнопку 109. inputpassword**[**counter**]** **=** pressedbtn**;** 110. counter**++;** 112. // Выводим на дисплей 113. writeLcdPassword**();** 114. delay**(**300**);** 116. // сохраняем сколько секунд прошло после запуска Arduino 117. pressedbtnsecs **=** seconds**();** 118. **}** 120. // если нажата кнопка "стереть" - стираем 121. **if** **((**pressedbtn **==** '-'**)** **&&** **(**counter **>** 0**))** 122. **{** 123. counter**--;** 124. inputpassword**[**counter**]** **=** '0'**;** 125. // чтобы не выйти за границу массива-пароля 126. **if** **(**counter **<** 0**)** 127. **{** 128. counter **=** 0**;** 129. **}** 131. // очищаем введённые символы на дисплее 132. writeLcdPassword**();** 133. delay**(**300**);** 135. // сохраняем сколько секунд прошло после запуска Arduino 136. pressedbtnsecs **=** seconds**();** 137. **}** 138. **}** 139. // Функция для определения какая кнопка нажата 140. char whatButton**()** 141. **{** 142. **if** **(**digitalRead**(**button1**)** **==** LOW**)** 143. **return** '1'**;** 144. **if** **(**digitalRead**(**button2**)** **==** LOW**)** 145. **return** '2'**;** 146. **if** **(**digitalRead**(**button3**)** **==** LOW**)** 147. **return** '3'**;** 148. **if** **(**digitalRead**(**button4**)** **==** LOW**)** 149. **return** '4'**;** 150. **if** **(**digitalRead**(**button5**)** **==** LOW**)** 151. **return** '5'**;** 152. **if** **(**digitalRead**(**button6**)** **==** LOW**)** 153. **return** '-'**;** // кнопка "стереть" 154. **return** '0'**;** // ничего не нажато 155. **}** 156. // Функция для проверки пароля 157. boolean checkPassword**()** 158. **{** 159. **for** **(**int i **=** 0**;** i **<** 5**;** i**++)** 160. **{** 161. **if** **(**password**[**i**]** **!=** inputpassword**[**i**])** 162. **return** **false;** // пароль не сошёлся 163. **}** 164. **return** **true;** // пароли одинаковы 165. **}** 166. // Функция для обнуления вводимого кода 167. void pwdtonull**()** 168. **{** 169. counter **=** 0**;** 170. **for(**int i **=** 0**;** i **<** 5**;** i**++)** 171. **{** 172. inputpassword**[**i**]** **=** '0'**;** 173. **}** 174. **}** 175. // Вывод введённого кода на дисплей 176. void writeLcdPassword**()** 177. **{** 179. **for** **(**int i **=** 0**;** i **<** 5**;** i**++)** 180. **{** 181. **if** **(**inputpassword**[**i**]** **==** '0'**)** 182. inputpassword**[**i**]** **=** ' '**;** // нули заменяем на пустоту 183. **}** 184. // Затираем всё что было до этого на дисплее 185. lcd**.**setCursor**(**0**,**1**);** 186. lcd**.**print**(**" "**);** 187. // И выводим введённые символы 188. lcd**.**setCursor**(**0**,**1**);** 189. lcd**.**print**(**inputpassword**);** 190. **}** 191. // Cекунды после запуска Arduino 192. int seconds**()** 193. **{** 194. **return** millis**()** **/** 1000**;** 195. **}** |

1. Разработка конструкции устройства

Разработка печатной платы производилась в программе Sprint-Layout 5, с учётом стандартных размеров элементов в размере 1:1.

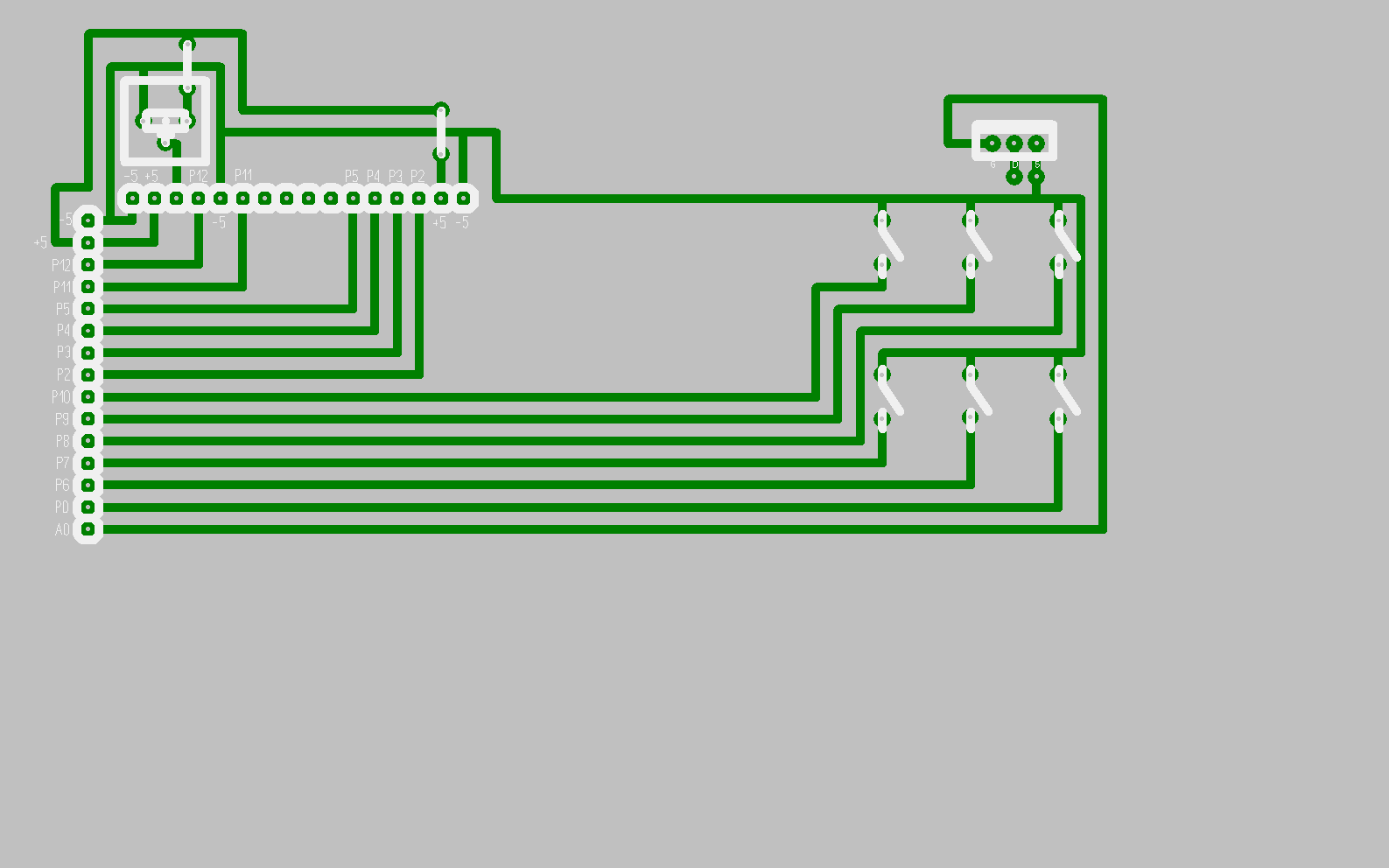


Рисунок 5.1. Чертёж печатной платы, выполненный в программе Sprint-Layout 5

Чертеж печатной платы приведен в приложении В. Чертеж платы с расположением элементов приведен в приложении Г.

Так же по данному чертежу было выполнено травление платы, с последующим монтированием компонентов. В итоге получилось полностью рабочее устройство.

Заключение

Итак, основная задача, а именно разработка электронного кодового замка выполнена. Разработан полностью рабочий вариант, который можно установить на дверь дома или на даче. Все пункты технического задания выполнены. В процессе проектирования были выполнены следующие действия:

* Изучен язык Wiring (AVR вариант С++) и принцип работы Arduino.
* Рассмотрены схемотехнические решения для реализации выбранного устройства. Следует сказать, что практически все схемы можно использовать для исполнения фильтра, каждое решение обладает как достоинствами, так и недостатками.
* Изучены существующие ГОСТы по оформлению принципиальных и печатных схем, по оформлению курсового проекта.
* Изучен метод травления печатной платы с помощью хлорного железа.
* Изучена программа Fritzing для прототипирования проектов на Arduino.
* Изучена программа Sprint-Layout 5 для изготовления чертежей печатных плат.

Литература

1. Охранные устройства и средства безопасности. *Сайт паяльник.* [В Интернете] http://cxem.net/guard/3.php
2. **Д., Нифашев.** Три схемы простейших кодовых замка (12В, на микросхеме) . *Радиотехнический сайт.* [В Интернете] http://www.dinistor.net.ru/tri-skhemy-prosteyshikh-kodovykh-zamka-12v-na-mikroskheme.html
3. **Ридико Л. И., Лапицкий В. П.** Электронный замок с ключами iButton. *Сайт паяльник.* [В Интернете] http://cxem.net/mc/mc15.php
4. Транзисторы. *Амперка.* [В Интернете] http://goo.gl/hJDpT
5. Программирование Arduino - введение. *RoboCraft.* [В Интернете] http://www.robocraft.ru/blog/arduino/29.html
6. Подключаем LCD-дисплей к Arduino. *RoboCraft.* [В Интернете] http://www.robocraft.ru/blog/arduino/503.html
7. Резисторы. *Амперка.* [В Интернете] http://goo.gl/55v8y
8. CraftDuino. *RoboCraft.* [В Интернете] http://robocraft.ru/blog/RoboCraft/97.html
9. Как подключить к микроконтроллеру кнопки. *Chip Enable.* [В Интернете] http://chipenable.ru/index.php/how-connection/13-connection-buttons-to-micros