|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Министерство науки и высшего образования Российской Федерации  Федеральное государственное бюджетное образовательное учреждение высшего образования  «Московский государственный технический университет имени Н.Э. Баумана»  (национальный технический университет)  Московский техникум космического приборостроения  УТВЕРЖДАЮ  Заместитель директора по УР  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.Н. Ковзель  (подпись, дата)   |  |  |  |  | | --- | --- | --- | --- | | **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**  **(ДИПЛОМНАЯ РАБОТА)**    **Программная реализация логического уровня автономных систем управления робототехнических устройств.**  Пояснительная записка  Группа ТИ-81 | | | | | Председатель предметной (цикловой) комиссии |  |  | Демина Е.С. | | (подпись, дата) |  | (Ф.И.О.) | | Руководитель разработки  от техникума |  |  | Демина Е.С. | | (подпись, дата) |  | (Ф.И.О.) | | Рецензент |  |  | Трефилов П.М. | | (подпись, дата) |  | (Ф.И.О.) | | Руководитель разработки  от предприятия |  |  | Шереужев М.А. | | (подпись, дата) |  | (Ф.И.О.) | | Студент |  |  | Коннов К.А. | | (подпись, дата) |  | (Ф.И.О.) |   Москва 2020 |

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана»

(национальный технический университет)

Московский техникум космического приборостроения

Срок окончания проекта «15» июня 2020 г.

Зав. отделением\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.Б. Ивлева Дата «22» апреля 2020 г.

*(подпись)*

ЗАДАНИЕ

на выполнение дипломной работы

Студенту Коннову Константину Александровичу

(фамилия, имя, отчество)

Тема работы Программная реализация логического уровня автономных систем управления робототехнических устройств.

Введение

1 Основная часть

* 1. Постановка задачи
  2. Описание логической структуры
  3. Описание метода организации данных
  4. Описание программно-технических средств
  5. Отладка программы
  6. Тестирование программы

1.7 Руководство пользователя

2 Охрана труда

2.1 Техника безопасности работы за персональным компьютером

Заключение

Список использованных источников

Приложения

Презентация

Дата выдачи задания «22» апреля 2020 г.

Руководитель дипломной работы от техникума\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (Ф. И.О.)

Руководитель работы от предприятия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (Ф. И.О.)

Срок сдачи обучающимся готового дипломной работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ г.

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) **ОТЗЫВ**

**руководителя на выпускную квалификационную работу**

Программная реализация логического уровня автономных систем управления робототехнических устройств.   
(тема выпускной квалификационной работы)

Выпускник Коннов Константин Александрович

(фамилия, имя, отчество)

Группа ТИ-81 Специальность 09.02.05 Прикладная информатика (по отраслям)

Выбор темы по согласованию с работодателем Шереужевым Мадином Артуровичом «Программная реализация логического уровня автономных систем управления робототехнических устройств.»

(согласованность – не согласованность темы), работодатель

Выпускная квалификационная работа (ВКР) была выполнена в учебном заведении МГТУ Н.Э. Баумана

Характеристика ВКР

напишите

Выводы: выпускная квалификационная работа рекомендована к защите по специальности 09.02.05 Прикладная информатика (по отраслям) Оценка:\_\_\_ваша оценка(ваша расшифровка)\_\_\_\_. Дипломант Коннов Константин Александрович заслуживает присвоения квалификации техник-программист.

Руководитель Шереужев Мадин Артурович

|  |  |  |
| --- | --- | --- |
| (фамилия, имя, отчество, ученая степень, должность) | (подпись) |  |
| «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г. |  |  |

СОДЕРЖАНИЕ

[ВВедение 5](#_Toc12190122)

[1.Основная часть 7](#_Toc12190123)

[1.1Постановка задачи 7](#_Toc12190124)

[1.1.1 Назначение прототипа 7](#_Toc12190125)

[1.1.2 Предметная область 7](#_Toc12190126)

[1.1.3 Требования к программе 7](#_Toc12190127)

[1.1.4 Требования к функциональным характеристикам 8](#_Toc12190128)

[1.2 Структурная схема программы 9](#_Toc12190129)

[1.2.1 Микроконтроллер 10](#_Toc12190130)

[1.2.2 Периферия 11](#_Toc12190131)

[1.2.3 Концепция программирования 13](#_Toc12190132)

[1.2.4 Язык программирования 13](#_Toc12190133)

[1.2.5 Загрузка программы в микроконтроллер 14](#_Toc12190134)

[1.2.6 Структура платы Arduino 15](#_Toc12190135)

[1.2.7 RFID 20](#_Toc12190136)

[1.2.8 Serial 25](#_Toc12190137)

[1.2.9 COM port 26](#_Toc12190138)

[1.2.10 UART 29](#_Toc12190139)

[1.2.11 Последовательный интерфейс UART в Arduino 30](#_Toc12190140)

[1.2.12 Архитектура микроконтроллеров AVR и PIC 32](#_Toc12190141)

[1.2.13 Библиотека Serial для работы с UART Arduino 34](#_Toc12190142)

[1.3 Описание метода организации данных 35](#_Toc12190143)

[1.3.1 Основные функции класса Serial 35](#_Toc12190144)

[1.3.2 Применение класса Serial 37](#_Toc12190145)

[1.4 Описание программно-технических средств 39](#_Toc12190146)

[1.4.1 Микроконтроллеры Arduino 39](#_Toc12190147)

[1.5 Отладка программы 41](#_Toc12190148)

[1.5.1 Применение библиотеки SPI.h 41](#_Toc12190149)

[1.6 Руководство пользователя 46](#_Toc12190150)

[2 ОХРАНА ТРУДА 48](#_Toc12190151)

[2.1 Техника безопасности при работе на компьютере 48](#_Toc12190152)

[2.1.1 Общие требования 48](#_Toc12190153)

[2.1.2 Требования безопасности перед началом работы 48](#_Toc12190154)

[2.1.3 Требования безопасности при выполнении работ 49](#_Toc12190155)

[2.1.4 Требования безопасности в аварийных ситуациях 50](#_Toc12190156)

[2.1.5 Требования безопасности по окончании работы 50](#_Toc12190157)

[2.2 Требования к помещению и оборудованию 51](#_Toc12190158)

[2.3 Мероприятия по противопожарной безопасности 53](#_Toc12190159)

[Заключение 55](#_Toc12190160)

[Список ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 56](#_Toc12190161)

[ПРИЛОЖЕНИЕ А 59](#_Toc12190162)

[ПРИЛОЖЕНИЕ Б 70](#_Toc12190163)

# ВВедение

Процесс проектирования цифровых устройств является очень долгой и трудоемкой задачей, для которой необходимы специалисты высокого уровня квалификации. Таким образом, обучение персонала является, пожалуй, наиболее важным аспектом для любой компании, занимающейся любой сферой деятельности, особенно разработкой различных устройств и программного обеспечения. Современный мир предоставляет множество способов для обучения – это школы, университеты, специализированные курсы и т.д. Сегодня один из наиболее популярных методов обучение – онлайн обучение. В интернете существуют множество ресурсов, предоставляющих как различную информацию, так и набор инструментов, необходимых для проведения экспериментов. Особенно это актуально для студентов таких направлений как компьютерная инженерия, поскольку им необходим опыт работы с реальными аппаратурными ресурсами для понимания принципов их работы. Одним из способов познакомиться с разработкой аппаратного обеспечения является использование так называемых программируемых логических интегральных схем (ПЛИС), которые представляют собой реконфигурируемый тип устройства.

В данной работе приведен один из основных этапов создания схемы ПЛИС, а именно разработка метода эмуляции периферийных устройств на базе Arduino.

Arduino – аппаратная вычислительная платформа, основными компонентами которой являются простая плата ввода/вывода и среда разработки на языке Processing/Wiring. Arduino может использоваться как для создания автономных интерактивных объектов, так и подключаться к программному обеспечению, выполняемому на компьютере (например, Macromedia Flash, Processing, Max/MSP, Pure Data, SuperCollider). Рассылаемые в настоящее время версии могут быть заказаны уже распаянными. Информация об устройстве платы (рисунок печатной платы) находится в открытом доступе и может быть использована теми, кто предпочитает собирать платы самостоятельно.

В дипломной работе рассмотрены основные принципы работы микроконтроллеров: Arduino IDE, UART, структура микроконтроллеров.

Во второй главе рассмотрены аналоги микроконтроллеров Arduino, принципы работы RFID и модули NFC.

**Объектом** выпускной квалификационной работы является процесс обработки различных сенсоров автономного робототехнического устройства, и последующая группировка данных в удобный вид.

**Предметом** выпускной квалификационной работы является рабочий образец на базе микроконтроллера OpenCR, двух инфракрасных сенсоров температуры mlx90614, двух моторов Dynamixel mx64, датчика цвета и двух web камер Logitech c920

**Целью работы** является разработка рабочего образца на базе сенсоров и микроконтроллера OpenCR для отправки данных в исполнительную часть робототехнического устройства.

Для достижения этой цели необходимо решить следующие задачи:

1. Настроить программные i2c адреса инфракрасных датчиков температуры;
2. Обработать данные с абсолютных энкодеров;
3. Обработать данные с датчика цвета;
4. Обработать изображение с web камер;
5. Отправка данных о состоянии всех датчиков
6. Отправка значений с датчиков

# Основная часть

## Постановка задачи

Тема дипломной работы – Программная реализация логического уровня автономных систем управления робототехнических устройств.

### 1.1.2 Предметная область

В ходе решения выполняемых задач, рабочий образец должен быть максимально надежным и компактным. Сенсоры должны возвращать правильные данные в условиях разной освещенности.

### 1.1.3 Требования к программе

Для работы выбраны следующие технические характеристики:

1. Микрокомпьютер OdroidC2 c установленной ОС Ubuntu server
2. Микроконтроллер OpenCR
3. Наличие двух свободных портов USB 2.0 и выше
4. Наличие двух ИК сенсоров mlx90614 с разными программными i2c адресами
5. Наличие двух web камер Logitech c920
6. Наличие датчика цвета
7. Наличие двух сервомоторов Dynamixel mx64

### 1.1.4 Требования к функциональным характеристикам

Разрабатываемый образец должен выполнять следующие функции:

1) Получение и отправка данных с энкодеров

2) Получение и отправка данных с датчиков температуры

3) Получение и отправка данных с датчика цвета

4) Обработка изображения из видеопотока и отправка сигнала при обнаружении искомого объекта

## 

## 1.2 Структурная схема программы

Структурная схема программы обработки сенсоров, представлена на рисунке 1.1.



Рисунок 1.1 – Структура программы «Main»

Вначале программы происходит инициализация сенсоров, затем программа заходит в бесконечный цикл, внутри которого данные с сенсоров получаются, проверяются и затем отправляются.

Структурная схема программы обработки изображений с веб камер, представлена на рисунке 1.2



Рисунок 1.2 Структура программы cam

## 1.2.1 Микроконтроллер

Микроконтроллеры для Arduino отличаются наличием предварительно прошитого в них загрузчика (bootloader). С помощью этого загрузчика пользователь загружает свою программу в микроконтроллер без использования традиционных отдельных аппаратных программаторов. Загрузчик соединяется с компьютером через интерфейс USB (если он есть на плате) или с помощью отдельного переходника UART-USB. Поддержка загрузчика встроена в Arduino IDE и выполняется в один щелчок мыши. На случай затирания загрузчика или покупки микроконтроллера без загрузчика разработчики предоставляют возможность прошить загрузчик в микроконтроллер самостоятельно.

Для этого в Arduino IDE встроена поддержка нескольких популярных дешевых программаторов, а большинство плат Arduino имеет штыревой разъем для внутрисхемного программирования (ICSP для AVR, JTAG для ARM). В Arduino IDE от компании, базирующейся на сайте arduino.cc, встроена возможность создания своих программно-аппаратных платформ. Этой возможностью пользуются сторонние компании, добавляющие в Arduino IDE свои наборы плат и компиляторов-загрузчиков к ним. Компания на сайте arduino.org не поддерживает такую возможность. AVR: В линейке устройств Arduino в основном применяются микроконтроллеры Atmel AVR ATmega328, ATmega168, ATmega2560, ATmega32U4, ATTiny85 с частотой тактирования 16 или 8 МГц. В старых изделиях применялись ATmega8, ATmega1280 и другие. ARM Cortex M: Есть также платы на процессоре ARM Cortex M. ESP8266: Сторонние разработчики портировали в Arduino поддержку популярного Wi-Fi микроконтроллера ESP8266. Теперь компилировать и загружать прошивку для ESP8266 со своими скетчами и поддержкой Wi-Fi можно прямо из Arduino IDE, получая одноплатную схему с поддержкой сети Wi-Fi. Intel x86: В рамках сотрудничества со сторонними производителями в Arduino IDE была включена поддержка некоторых аппаратных средств Intel x86. Intel Galileo (процессор Intel Quark X1000 400 МГц), Intel Edison и Arduino 101 - Arduino-совместимые платы на Intel x86 архитектуре. Платы механически и электрически совместимы с периферийными платами Arduino. Платы функционируют под собственной ОС Linux, поверх которой работает приложение, позволяющее загружать и исполнять скетчи Arduino.

## 1.2.2 Периферия

Порты ввода-вывода микроконтроллеров оформлены в виде штыревых линеек. Микроконтроллеры питаются от 5В или 3,3В, в зависимости от модели платы. Соответственно, порты имеют такой же размах допустимых входных и выходных напряжений. Программисту доступны некоторые специальные возможности портов ввода-вывода микроконтроллеров, например широтно-импульсная модуляция (ШИМ), аналогово-цифровой преобразователь (АЦП), интерфейсы UART, SPI, I2C. Количество и возможности портов ввода-вывода определяются конкретным вариантом микропроцессорной платы.

Помимо портов на платах микроконтроллеров иногда устанавливается периферия в виде интерфейсов USB или Ethernet. Опциональный набор внешней периферии на модулях расширения включает в себя: ·USB Device (чаще всего как виртуальный COM порт через FTDI FT232, имеются также версии с эмуляцией USB HID Class клавиатур и мышек); ·Модуль GSM и другие беспроводные интерфейсы;·USB Host;·SD card;·Модуль управления низковольтным мотором на базе L298. Поддерживаются шаговый и коллекторный двигатели с напряжением до 12В и током до 2А на канал. Могут подключаться также реле, электромагниты и т. п. Модуль не имеет гальванической развязки; ·Графический ЖКИ индикатор; ·Модуль с макетным полем.

Serial Arduino, программируется через последовательное соединение (разъём DB-9), используется ATmega8. Arduino Extreme, с USB-интерфейсом для программирования, используется ATmega8. Arduino Mini, миниатюрная версия Arduino, использующая поверхностный монтаж ATmega328. Не содержит конвертера USB-UART. Arduino Nano 3.0, ещё миниатюрнее, с питанием от USB и поверхностным монтажом ATmega328. LilyPad Arduino, минималистичный дизайн для носимых применений с поверхностным монтажом ATmega168 (в новых версиях ATmega328). Arduino NG, с USB-интерфейсом для программирования, используется ATmega8. Arduino NG plus, с USB-интерфейсом для программирования, используется ATmega168. Arduino BT, с Bluetooth-интерфейсом для программирования, используется ATmega168 (в новых версиях ATmega328). Arduino Diecimila, использует USB-интерфейс и Atmega168 в DIP28 корпусе. .Arduino Duemilanove («2009»), на основе ATmega168 (в новых версиях ATmega328), с автоматическим выбором питания от USB или внешнего источника. Arduino Mega («2009»), на основе ATmega1280. Arduino Mega2560 R3 («2011»), на основе ATmega2560. Используется конвертер USB-UART на базе ATmega16U2. Arduino Uno R3 (2011), на основе ATmega328. Используется конвертер USB-UART на базе ATmega16U2, в нашей работе используем именно его. 14.Arduino Ethernet (2011), на основе ATmega328. Конвертера USB-UART нет. Ethernet чип - W5100, также содержит модуль MicroSD. .Arduino Mega ADK for Android (2011), на основе ATmega2560. Содержит USB-хост для соединения с телефонами на базе ОС Android (м/с MAX3421e). Конвертер USB-UART на базе ATmega32U2. Ниже на рисунке 1.2 и 1.3 приведены примеры микроконтроллеров на базе Atmega 32U4.

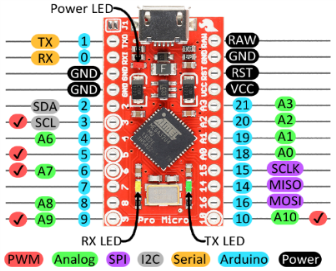


Рисунок 1.2 – Arduino pro micro

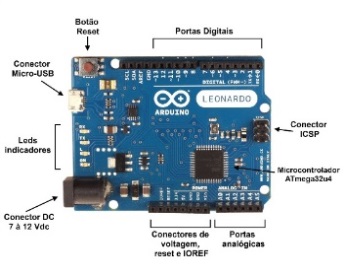


Рисунок 1.3 - Arduino Leonardo

## 1.2.3 Концепция программирования

Программирование ведется целиком через собственную программную оболочку (IDE), бесплатно доступную на сайте Arduino. В этой оболочке имеется текстовый редактор, менеджер проектов, препроцессор, компилятор и инструменты для загрузки программы в микроконтроллер. Оболочка написана на Java на основе проекта Processing, работает под Windows, Mac

OS X и Linux.

## 1.2.4 Язык программирования

Язык программирования Arduino является стандартным C++ (используется компилятор AVR-GCC) с некоторыми особенностями, облегчающими новичкам написание первой работающей программы. Программы, написанные программистом Arduino называются наброски (или иногда скетчи - варваризм от англ. sketch) и сохраняются в файлах с расширением ino. Эти файлы перед компиляцией обрабатываются препроцессором Arduino. Также существует возможность создавать и подключать к проекту стандартные файлы C++. Обязательную в C++ функцию main() препроцессор Arduino создает сам, вставляя туда необходимые «черновые» действия. Программист должен написать две обязательные для Arduino функции setup() и loop(). Первая вызывается однократно при старте, вторая выполняется в бесконечном цикле. В текст своей программы (скетча) программист не обязан вставлять заголовочные файлы используемых стандартных библиотек. Эти заголовочные файлы добавит препроцессор Arduino в соответствии с конфигурацией проекта. Однако пользовательские библиотеки нужно указывать. Менеджер проекта Arduino IDE имеет нестандартный механизм добавления библиотек. Библиотеки в виде исходных текстов на стандартном C++ добавляются в специальную папку в рабочем каталоге IDE. При этом название библиотеки добавляется в список библиотек в меню IDE. Программист отмечает нужные библиотеки и они вносятся в список компиляции.IDE не предлагает никаких настроек компилятора и минимизирует другие настройки, что упрощает начало работы для новичков и уменьшает риск возникновения проблем.

Простейшая Arduino-программа состоит из двух функций: ·setup(): функция вызывается однократно при старте микроконтроллера. ·loop(): функция вызывается после setup () в бесконечном цикле все время работы микроконтроллера. Так выглядит полный текст простейшей программы (скетча) мигания светодиодом с периодом 2 секунды: #define LED\_PIN 13 //Связывание 13-го выхода платы с переменной LED\_PINsetup () {(LED\_PIN, OUTPUT); // Назначение переменной LED\_PIN и 13-го выхода на вывод }loop () {(LED\_PIN, HIGH); // Включение светодиода, параметр вызова функции digitalWrite HIGH - признак высокого логического уровня(1000); // Цикл задержки на 1000 мс – 1 секунду(LED\_PIN, LOW); // Выключение светодиода, параметр вызова LOW – признак низкого логического уровня(1000); // Цикл задержки на 1 секунду } Все используемые в примере функции являются библиотечными. В комплекте Arduino IDE имеется множество примеров программ. Существует перевод документации по Arduino на русский язык.

## 1.2.5 Загрузка программы в микроконтроллер

Закачка программы в микроконтроллер Arduino происходит через предварительно запрограммированный специальный загрузчик (все микроконтроллеры от Arduino продаются с этим загрузчиком). Загрузчик создан на основе Atmel AVR Application Note AN109. Загрузчик может работать через интерфейсы RS-232, USB или Ethernet в зависимости от состава периферии конкретной процессорной платы. В некоторых вариантах, таких как Arduino Mini или неофициальной Boarduino, для программирования требуется отдельный переходник. Пользователь может самостоятельно запрограммировать загрузчик в чистый микроконтроллер. Для этого в IDE интегрирована поддержка программатора на основе проекта AVRDude. Поддерживается несколько типов популярных дешёвых программаторов.

## 1.2.6 Структура платы Arduino

В данном разделе будут рассмотрены компоненты Arduino двигаясь согласно диаграмме по часовой стрелке, начиная с разъёма USB-B. Ниже на рисунке 1.4 изображена диаграмма компонентов Arduino.

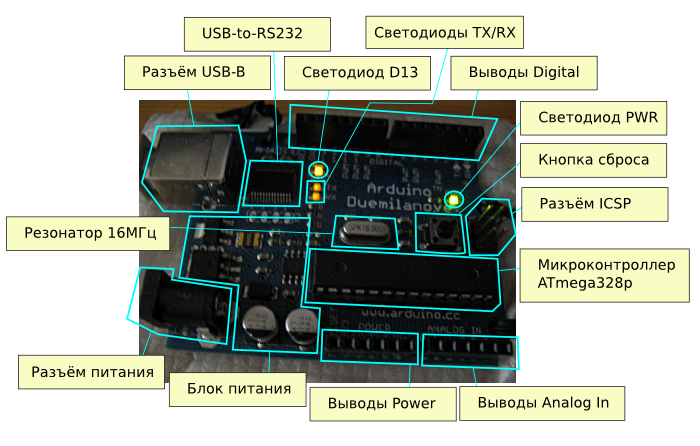


Рисунок 1.4 – Диаграмма компонентов Arduino

Разъём USB-B.

Коннектор шины USB, в данном случае "папа" типа B. Предназначен для подключения Arduino по шине USB в качестве slave-устройства к различным USB-хостам (например, к компьютеру). На рисунке 1.5 изображена разъем USB-B

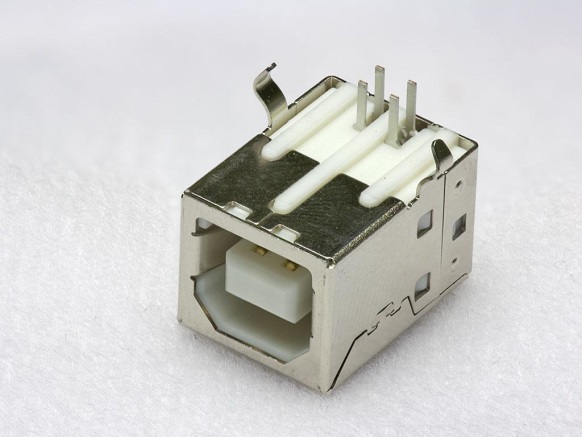


Рисунок 1.5 – Разъем USB-B

USB-to-USART.

Микросхема-преобразователь, позволяющая организовать линию передачи данных формата USART по шине USB. На рисунке 1.6 изображен USB-to-USART преобразователь.

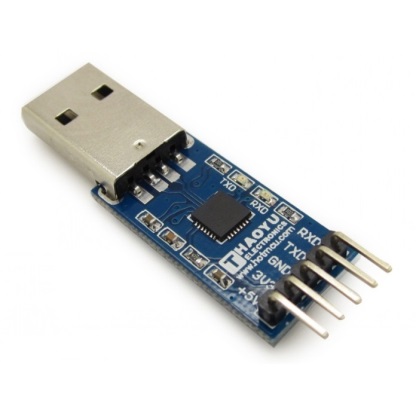


Рисунок 1.6 – USB-to-USART преобразователь

На разных платах Arduino эти микросхемы различаются. Например, в Duemilanove (на фото) используется чип FTDI USB<->USART, а например в Uno и Mega 2560 это преобразование выполняет специально запрограммированный микроконтроллер ATmega8U2.

Светодиод D13

Светодиод, подключенный через токоограничительный резистор к выводу Digital 13. Можно сказать, что это главный мигательный светодиод Arduino, во всяком случае на моей Duemilanove скетч примера Blink мигает именно этим светодиодом.

Светодиоды индикации Transmitter Exchages и Receiver Exchanges, подключенные к выводам чипа USB-to-RS232. Когда светится светодиод TX, это означает, что Arduino передаёт данные хосту, и наоборот - при приёме данных Arduino от хоста светится светодиод RX. Таким образом маркировка TX означает "я передаю (Transmit) данные", а RX означает "я принимаю (Receive) данные". На рисунке 1.7 изображена плата Arduino с распаянными светодиодами RX/TX.

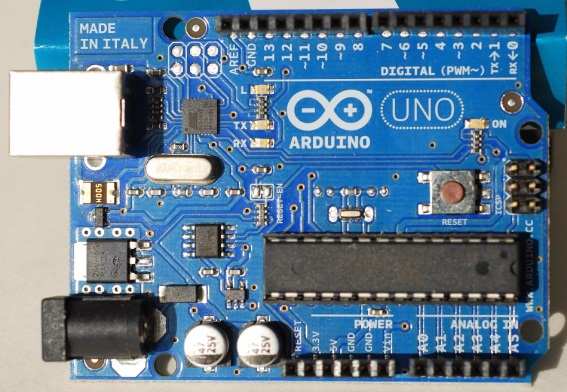


Рисунок 1.7 – Плата Arduino UNO с распаянными светодиодами RX/TX

Выводы Digital.

Тринадцать цифровых контактов портов ввода-вывода D и B микроконтроллера ATmega328p, к которым добавлены контакты GND (земля) и AREF (Analog Reference Voltage -- Аналоговое Сравнительное Напряжение), сгруппированные в две гребёнки типа 8x1 с шагом 2.54 мм (0.1 дюйма). На рисунке 1.8 изображена плата Arduino с выводами Digital.

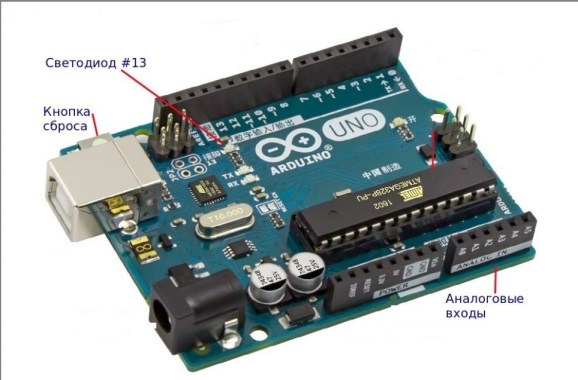


Рисунок 1.8 – Плата Arduino UNO с выводами Digital

Выводы Digital 0..7 соответствуют порту ввода-вывода D, а выводы Digital 8..13 младшим пяти разрядам порта ввода-вывода B.

Кнопка сброса.

При нажатии и отпускании этой кнопки с говорящей подписью RESET (на Mega 2560) или ничего не говорящей подписью S1 (на Duemilanove) производится перезапуск микроконтроллера. На рисунке 1.9 изображена кнопка reset на плате Arduino UNO.

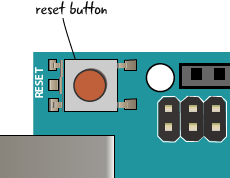


Рисунок 1.9 – Кнопка RESET на плате Arduino UNO

При удержании низкого сигнала микроконтроллер можно перепрограммировать через выводы разъёма ICSP благодаря встроенной функции ISP (In-System Programming). На рисунке 1.10 изображены разъемы SPI на плате Arduino UNO.

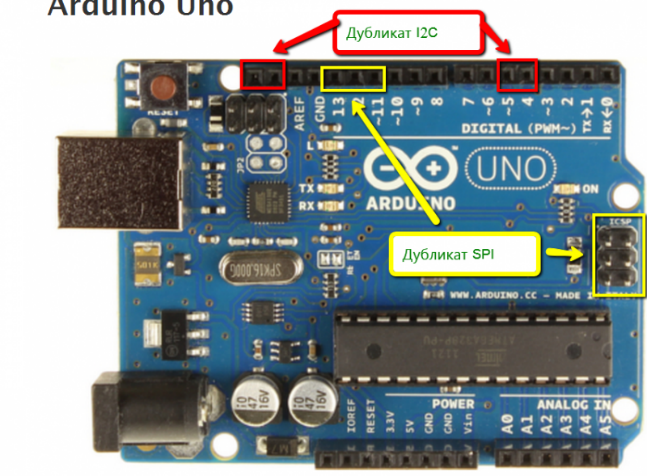


Рисунок 1.10 –Разъемы SPI на плате Arduino UNO

Шесть выводов разъёма ICSP представляют собой интерфейс SPI за исключением сигнала SS (Slave Select -- Выбор Исполнительного Устройства). Посредством данного интерфейса можно программировать микроконтроллер Arduino, минуя загрузчик bootloader, либо если требуется прошить новую версию загрузчика или управляющие биты FUSE.

Микроконтроллер ATmega328p.

Ядром Arduino как платы является микроконтроллер, в случае с Duemilanove это ATmega328p. Подробнее о том, как устроен сам микроконтроллер, можно прочитать в статье Микроконтроллер Arduino. На рисунке 1.11 изображен микроконтроллер ATmega328p.

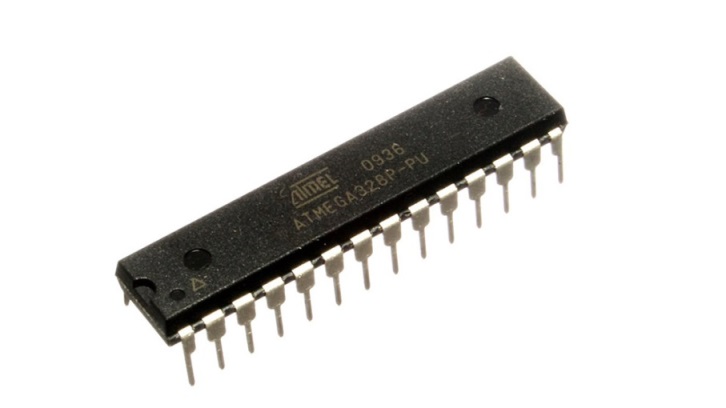


Рисунок 1.11 –Микроконтроллер ATmega328p

Выводы Analog In.

Младшие шесть разрядов порта ввода-вывода микроконтроллера ATmega328p сгруппированы в контактную группу Analog In.

Название группы (по-русски Аналоговый Ввод) этих контактов может смутить, но дело в том, что эти выводы микроконтроллера совмещены со входами каналов аналого-цифрового преобразователя (АЦП) Arduino, так что любой из них может использоваться либо как обычный цифровой пин ввода-вывода, либо как входной вывод канала АЦП. Поэтому они промаркированы на плате как Analog In, что несколько сбивает с толку начинающего - можно ли их использовать для цифрового вывода.

Для цифрового можно, аналоговый - только ввод.

Выводы Power.

В этой контактной гребёнке сгруппированы напряжения питания, земля и сигнал сброса.

Сигнал +3.3 Вольт можно использовать только как reference voltage, поскольку происходит это напряжение с reference-вывода чипа FTDI USB<->RS-232. И хотя на платах c USB через ATmega8U2 для совместимости ставится LDO-регулятор, который способен обеспечить ток в 0.15 Ампер, хорошей практикой для шилдов Arduino является именно взять с Arduino +5 Вольт и через регулятор получить питающие +3.3 Вольт.

Сигнал сброса является не только выходом, но и входом, плату можно сбрасывать, используя этот контакт. С остальными контактами ситуация следующая:

* Vin   - ввод и вывод;
* Gnd   - земля;
* 5V    -- вывод; ввод только в случае, если не требуется гарантировать бесперебойное выполнение скетча в ситуации, когда USB было подключено, а затем снова отключено;
* 3V3   -- вывод;
* RESET -- ввод и вывод.

Блок питания.

Почти четверть площади платы занимают элементы блока питания Arduino. Но популярность решения такова, что скорее всего чип где-то кем-то, возможно просто запатентован неудачно.

Разъём питания

Максимальное входное напряжение, подаваемое на этот разъём или контакт Vin, равно 20 Вольтам. Не забыть добавить диод.

Резонатор 16МГц.

Кварцевый резонатор 16МГц, выравнивающий частоту импульсов основной тактовой частоты микроконтроллера. Читал я в Сети, что его можно выпаять, впаять кристалл на 20 МГц, перепрошить загрузчик bootloader, и что вроде как работает, даже не греется. Но увы, дело тут не в перегреве, а во внутреннем исполнении микросхемы. Так что на +5В оно возможно и работает, а если батарейка сядет до +4В, будет ли работать -- вот в чём печаль.

## 1.2.7 RFID

RFID (англ. *Radio Frequency IDentification*, радиочастотная идентификация) — способ автоматической идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в так называемых транспондерах, или RFID-метках.

Любая RFID-система состоит из считывающего устройства (считыватель, ридер или интеррогатор) и транспондера (он же RFID-метка, иногда также применяется термин RFID-тег).

По дальности считывания RFID-системы можно подразделить на системы:

* ближней идентификации (считывание производится на расстоянии до 20 см);
* идентификации средней дальности (от 20 см до 5 м);
* дальней идентификации (от 5 м до 300 м).

Большинство RFID-меток состоит из двух частей. Первая — интегральная схема (ИС) для хранения и обработки информации, модулирования и демодулирования радиочастотного (RF) сигнала и некоторых других функций. Вторая — антенна для приёма и передачи сигнала.

C введением RFID-меток в повседневную жизнь связан ряд проблем. Например, потребители, не обладающие считывателями, не всегда могут обнаружить метки, прикреплённые к товару на этапе производства и упаковки, и избавиться от них. Хотя при продаже, как правило, такие метки уничтожаются, сам факт их наличия вызывает опасения у правозащитных организаций и некоторых представителей Русской Православной Церкви.

Уже известные приложения RFID (бесконтактные карты в системах контроля и управления доступом, системах дальней идентификации и в платёжных системах) получают дополнительную популярность с развитием интернет-услуг.

Классификация RFID-меток.

Существует несколько способов систематизации RFID-меток и систем:

* по рабочей частоте;
* по источнику питания;
* по типу памяти;
* по исполнению;
* по источнику питания.

По типу источника питания RFID-метки делятся на:

* пассивные;
* активные;
* полупассивные.

Пассивные.

На рисунке 1.12 показана RFID антенна.

[](https://commons.wikimedia.org/wiki/File:Rfid-antenna.JPG?uselang=ru)

Рисунок 1.12— RFID антенна

Пассивные RFID-метки не имеют встроенного источника энергии. Электрический ток, индуцированный в антенне электромагнитным сигналом от считывателя, обеспечивает достаточную мощность для функционирования кремниевого КМОП-чипа, размещённого в метке, и передачи ответного сигнала. На рисунке 1.13 изображена пассивная RFID метка.

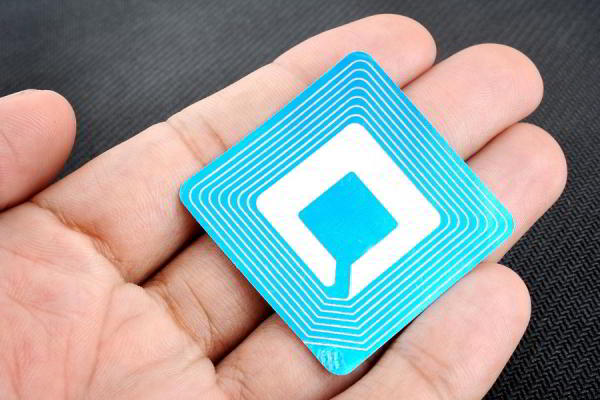


Рисунок 1.13 — Пассивная RFID метка

Каждая метка имеет идентификационный номер. Пассивные метки могут содержать перезаписываемую энергонезависимую память EEPROM-типа. Дальность действия меток составляет 1—200 см (ВЧ-метки) и 1-10 метров (УВЧ и СВЧ-метки).

Активные.

Активные RFID-метки обладают собственным источником питания и не зависят от энергии считывателя, вследствие чего они читаются на дальнем расстоянии, имеют большие размеры и могут быть оснащены дополнительной электроникой. Однако, такие метки наиболее дороги, а у батарей ограничено время работы. На рисунке 1.14 показана активная RFID метка.



Рисунок 1.14— Активные RFID метки

Полупассивные.

Полупассивные RFID-метки, также называемые полуактивными, очень похожи на пассивные метки, но оснащены батареей, которая обеспечивает чип энергопитанием. При этом дальность действия этих меток зависит только от чувствительности приёмника считывателя, и они могут функционировать на большем расстоянии и с лучшими характеристиками.

По типу используемой памяти.

По типу используемой памяти RFID-метки делятся на:

* RO (англ. *Read Only*) — данные записываются только один раз, сразу при изготовлении. Такие метки пригодны только для идентификации. Никакую новую информацию в них записать нельзя, и их практически невозможно подделать.
* WORM (англ. *Write Once Read Many*) — кроме уникального идентификатора такие метки содержат блок однократно записываемой памяти, которую в дальнейшем можно многократно читать.
* RW (англ. *Read and Write*) — такие метки содержат идентификатор и блок памяти для чтения/записи информации. Данные в них могут быть перезаписаны многократно.

Радиочастотные UHF-метки ближнего поля.

Метки ближнего поля (англ. *UHF Near-Field*), не являясь непосредственно радиометками, а используя магнитное поле антенны, позволяют решить проблему считывания в условиях высокой влажности, присутствия воды и металла. С помощью данной технологии ожидается начало массового применения RFID-меток в розничной торговле фармацевтическими товарами (нуждающимися в контроле подлинности, учёте, но при этом зачастую содержащими воду и металлические детали в упаковке).

Ридеры (Считыватели).

Приборы, которые читают информацию с меток и записывают в них данные. Эти устройства могут быть постоянно подключенными к учётной системе или работать автономно.

Виды считывателей

На рисунке 1.15 показан настольный RFID-считыватель.

[](https://commons.wikimedia.org/wiki/File:Rfid-reader(table).JPG?uselang=ru)

Рисунок 1.15— Настольный RFID считыватель

Стационарные.

Стационарные считыватели крепятся неподвижно на стенах, дверях, движущихся складских устройствах.

Мобильные.

Обладают сравнительно меньшей дальностью действия и зачастую не имеют постоянной связи с программой контроля и учёта. Мобильные считыватели имеют внутреннюю память, в которую записываются данные с прочитанных меток (потом эту информацию можно загрузить в компьютер) и, как и стационарные считыватели, способны записывать данные в метку (например, информацию о произведённом контроле). На рисунке 1.16 показан мобильный RFID считыватель.



Рисунок 1.16— Мобильный RFID считыватель

В зависимости от частотного диапазона метки, дистанция устойчивого считывания и записи данных в них будет различна.

RFID и альтернативные методы автоматической идентификации. На рисунке 1.17 изображена RFID-метка SIMATIC RF620T.



Рисунок 1.17— RFID-метка SIMATIC RF620T

## 1.2.8 Serial

Набор функций **Serial** служит для связи устройства Arduino с компьютером или другими устройствами, поддерживающими последовательный интерфейс обмена данными. Все платы Arduino имеют хотя бы один последовательный порт ([UART](http://ru.wikipedia.org/wiki/UART), иногда называют USART). Для обмена данными **Serial** используют цифровые порты ввод/вывода 0 (RX) и 1 (TX), а также USB порт. Важно учитывать, что если вы используете функции **Serial**, то нельзя одновременно с этим использовать порты 0 и 1 для других целей. [Среда разработки Arduino](http://arduino.ru/Arduino_environment) имеет встроенный монитор последовательного интерфейса (Serial monitor). Для начала обмена данными необходимо запустить монитор нажатием кнопки Serial monitor и выставить ту же скорость связи (baud rate), с которой вызвана функция begin().

Плата [Arduino Mega](http://arduino.ru/Hardware/ArduinoBoardMega) имеет три дополнительных последовательных порта: **Serial1** на портах 19 (RX) и 18 (TX), **Serial2** на портах на портах 17 (RX) и 16 (TX), **Serial3** на портах на портах 15 (RX) и 14 (TX). Чтобы использовать эти порты для связи с компьютером понадобится дополнительные адаптеры USB-to-serial, т.к. они не подключены к встроенному адаптеру платы Mega. Для связи с внешним устройством через последовательный интерфейс соедините TX порт вашего устройства с RX портом внешнего устройства и RX порт вашего устройства с портом TX внешнего и соедините "землю" на устройствах.

## 1.2.9 COM port

Все современные компьютеры комплектуются портами нового поколения — USB. Это порты последовательного типа, но с высокой производительностью (до 12 Мбайт/с). Кроме высокой производительности к достоинствам USB портов относится удобство работы с ними: не требуется выключать оборудование перед стыковкой, возможно подключение нескольких устройств в одному порту. Многие модели современной периферийного оборудования могут подключаться к портам этого типа.

Кроме универсальных коммуникационных портов, предназначенных для любого оборудования, компьютер имеет два специализированных порта для подключения мыши и клавиатуры — это порты PS/2. Другие устройства к этим портам не подключаются.

Первые IBM PC предоставляли:

* встроенный порт для подключения клавиатуры;
* до 4-х (COM1 … COM4) последовательных портов, обычно служащих для подключения, сравнительно высокоскоростных, коммуникационных устройств, использующих интерфейс RS-232, например модемов. Для них выделялись следующие ресурсы материнской платы:
* базовые порты ввода-вывода: 3F0.3FF (COM1), 2F0.2FF (COM2), 3E0.3EF (COM3) и 2E0.2EF (COM2)
* номер IRQ: 3 (COM2/4), 4 (COM1/3);
* до 3-х (LPT1. LPT3) параллельных портов (англ. Line Print Terminal), обычно служащих для подключения принтеров использующих интерфейс IEEE 1284. Для них выделялись следующие ресурсы материнской платы:
* базовые порты ввода-вывода: 370.37 °F (LPT1 или LPT2 только в компьютерах IBM с MCA), 270.27 °F (LTP2 или LPT3 только в компьютерах IBM с MCA] и 3B0.3BF (LPT1 только в компьютерах IBM с MCA)

Изначально, COM и LPT порты на материнской плате отсутствовали физически и реализовались дополнительной картой расширения, вставляемой в один из ISA-слотов расширения на материнской плате.

Последовательные порты как правило использовались для подключения устройств, которым требовалась быстро передать небольшой объём данных, например компьютерной мыши и внешнего модема, а параллельные -- для принтера или сканера, для которых передача большого объёма не была критичной по времени. В дальнейшем, поддержка последовательных и параллельных портов была интегрирована в чипсеты, реализующие логику материнской платы.

Недостаток интерфейсов RS-232 и IEEE 1284 -- относительно малая скорость передачи данных, не удовлетворяющая растущие потребности в передаче данных между устройствами. Как следствие, появились новые стандарты интерфейсных шин USB.

Особенность USB является то, что при подключении многих USB-устройств к единственному USB-порту используют т. н. концентраторы (USB-хабы), которые в свою очередь коммутируют между собой, увеличивая тем самым число USB-устройств, которые можно подключать. Такая топология шины USB называется «звезда» и включает в себя также корневой концентратор, который, как правило, находится в «южном мосте» материнской платы компьютера, к которому и подключаются все дочерние концентраторы (в частном случае сами USB-устройства). В таблице 1.1 сравнение интерфейсов COM, LTP, USB.

Таблица 1.1 – Сравнение интерфейсов COM, LTP, USB.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Интерфейс | Количество поддерживаемых устройств | Пропускная способность | Возможность подключения по цепочке | Макс. длина кабеля |
| COM | 1 | 115,2 Кбит/c | НЕТ | 15−20 м |
| LPT | 1 | 600 Кбит - 1,5 Мбит/c | НЕТ | 4 м |
| USB | 127 | 1,5 Мбит/c - 5 Гбит/с | ДА | 4 м |

**Последовательный порт** (англ. Serial port), серийный порт или **COM-порт** - двунаправленный последовательный интерфейс, предназначенный для обмена битовой информацией.

Последовательным данный порт называется потому, что информация через него передаётся по одному биту, бит за битом (в отличие от параллельного порта). Хотя некоторые другие интерфейсы компьютера -- такие как Ethernet, FireWire и USB -- также используют последовательный способ обмена, название «последовательный порт» закрепилось за портом, имеющим стандарт RS-232C.

Наиболее часто для последовательного порта персональных компьютеров используется стандарт RS-232C. Ранее последовательный порт использовался для подключения терминала, позже для модема или мыши. Сейчас он используется для соединения с источниками бесперебойного питания, для связи с аппаратными средствами разработки встраиваемых вычислительных систем, спутниковыми ресиверами, а также с приборами систем безопасности объектов.

С помощью COM-порта можно соединить два компьютера, используя так называемый «нуль-модемный кабель».

На материнских платах ведущих производителей (например, Intel) или готовых системах (например, IBM, Hewlett-Packard, Fujitsu Siemens Computers) для последовательного порта принято следующее условное обозначение: Наиболее часто используются стандартизированные в 1969 году D-образные разъёмы: 9-ти и 25-тиконтактные, (DE-9 и DB-25 соответственно). Раньше использовались также DB-31 и круглые восьмиконтактные DIN-8.

## 1.2.10 UART

Почти каждый микроконтроллер имеет на борту универсальный последовательный интерфейс – UART. AVR тут не исключение и поддерживает этот протокол в полном объеме полностью аппаратно. По структуре это обычный асинхронный последовательный протокол, то есть передающая сторона по очереди выдает в линию 0 и 1, а принимающая отслеживает их и запоминает. Синхронизация идет по времени – приемник и передатчик заранее договариваются о том на какой частоте будет идти обмен.  
 Вначале передатчик бросает линию в низкий уровень – это старт бит. Почуяв что, линия просела, приемник выжидает интервал Т1 и считывает первый бит, потом через интервалы Т2 выковыриваются остальные биты. Последний бит это стоп бит. Говорящий о том, что передача этого байта завершена. Это в самом простом случае. В конце байта, перед стоп битом, может быть и бит четности. Который получается если поксорить между собой все биты, для контроля качества передачи. Также может быть два стопа, опять же для надежности. Битов может быть не 8, а 9. О всех этих параметрах договариваются на берегу, до начала передачи. Самым же популярным является 8 бит, один старт один стоп, без четности. На рисунке 1.18 представлена побитовая передача информации в интерфейсе UART.

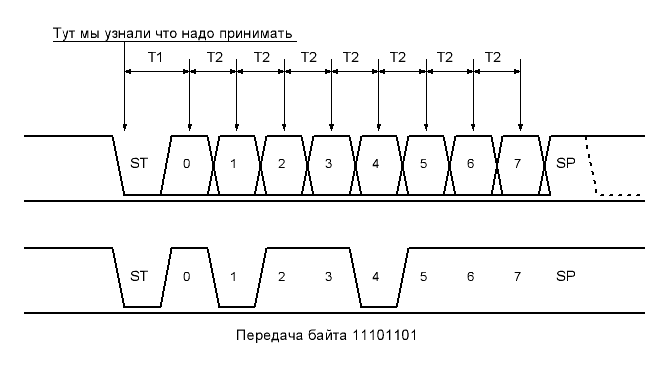


Рисунок 1.18 – Интерфейс UART. Передача байта 11101101

По такому протоколу работает COM порт компьютера, отличие в разнице напряжений, поэтому именно этот протокол используется для связи микроконтроллера с персональным компьютером (ПК). Для преобразования напряжений [можно использовать RS232-TTL конвертер.](http://easyelectronics.ru/svyaz-mikrokontrollera-s-kompyuterom-cherez-rs232.html) Мы же применим встроенный в [Pinboard](http://shop.easyelectronics.ru/index.php?productID=147) мост USB-UART который образовывает в системе виртуальный COM PORT.

На платах [Arduino](http://iarduino.ru/shop/arduino/) c USB портом (у некоторых плат нет USB порта) уже есть USB-UART преобразователь на базе чипа CH340 или FTDI, или ATmega, или др. Шина USB этих чипов подключена к порту USB, а шина UART к аппаратным выводам TX и RX контроллера платы [Arduino](http://iarduino.ru/shop/arduino/).

Всё что нужно сделать для работы платы [Arduino](http://iarduino.ru/shop/arduino/) в качестве USB-UART преобразователя, это отключить её контроллер от шины UART или заставить его не реагировать на сигналы проходящие по этой шине. Сделать это можно соединив вывод RESET с выводом GND платы [Arduino](http://iarduino.ru/shop/arduino/).

Теперь плату [Arduino](http://iarduino.ru/shop/arduino/) можно подключить к USB порту компьютера, а внешние устройства к выводам TX и RX платы [Arduino](http://iarduino.ru/shop/arduino/) (обычно это выводы 0 и 1). Обратите внимание на то, что выводы устройств подключаются к плате, не TX к RX, RX к TX, а TX к TX, RX к RX (так как вывод TX чипа преобразователя на плате [Arduino](http://iarduino.ru/shop/arduino/) подключён к выводу RX самой платы, а вывод RX чипа подключён к выводу TX платы).

Для управления подключённым устройством нужно установить на компьютер программу терминал. Одним из таких терминалов является свободно распространяемая программа [Termite](http://www.compuphase.com/software_termite.htm) с поддержкой Русского языка.

## 1.2.11 **Последовательный интерфейс UART в Arduino**

Любая плата Arduino имеет, как минимум, один аппаратный последовательный интерфейс UART. Платы Arduino Mega и Arduino Due имеют по три порта.

В контроллере существует электронный узел, в  регистр которого программа только загружает байт для передачи, а формирование сигналов обмена и все остальные операции делает этот узел. Может быть реализована и программная передача данных по протоколу UART. В этом случае все сигналы формируются программой. Конечно, это занимает ресурсы процессора.

Плата Arduino UNO имеет один порт UART, сигналы которого  подключены к выводам 0 (сигнал RX) и 1 (сигнал TX). Сигналы имеют логические уровни TTL (0…5 В). Через эти выводы (0 и 1) можно подключить к плате другое устройство имеющее интерфейс UART.

Кроме функции связи с другими контроллерами порт UART платы Arduino UNO используется для загрузки в контроллер программы из компьютера. Для этого к этим же сигналам (RX и TX) подключены соответствующие выводы микросхемы ATmega16U2 – преобразователя интерфейса USB/UART. Микросхема преобразователя подключена через резисторы сопротивлением 1 кОм. Ниже на рисунке 1.19 представлен фрагмент схемы платы Arduino UNO R3.

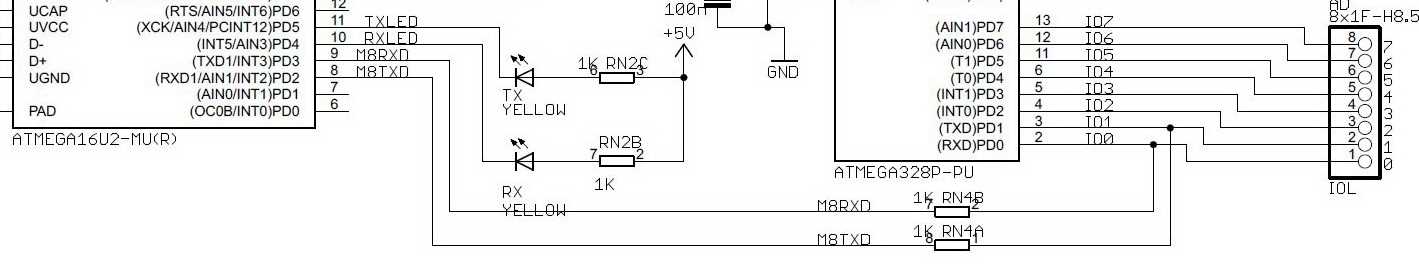
****

Рисунок 1.19 – Фрагмент схемы платы Arduino UNO R3

Таким образом, при свободных выводах 0 и 1 платы Arduino сигналы с микросхемы ATmega16U2 поступают на контроллер ATmega328. При подключении внешнего UART устройства, то сигналы будут иметь приоритет, т.к. ATmega16U2 подключена через резисторы.

Преобразователь интерфейса ATmega16U2 позволяет подключать плату Arduino к компьютеру через USB порт. На компьютер устанавливается драйвер. Он создает на компьютере виртуальный COM порт. Через него и происходит обмен. Такая технология описана на примере другого преобразователя интерфейсов PL2303 USB-UART.

Важно понимать несмотря на то, что плата подключена к компьютеру через USB порт, все программы обмениваются данными через виртуальный COM порт, не подозревая, что порт виртуальный.

## 1.2.12 Архитектура микроконтроллеров AVR и PIC

Все микроконтроллеры построены по одной схеме. Система управления, состоящая из счетчика команд и схемы декодирования, выполняет считывание и декодирование команд из памяти программ, а операционное устройство отвечает за выполнение арифметических и логических операций; интерфейс ввода/вывода позволяет обмениваться данными с периферийными устройствами; и, наконец, необходимо иметь запоминающее устройство для хранения программ и данных, показано на рисунке 1.20.

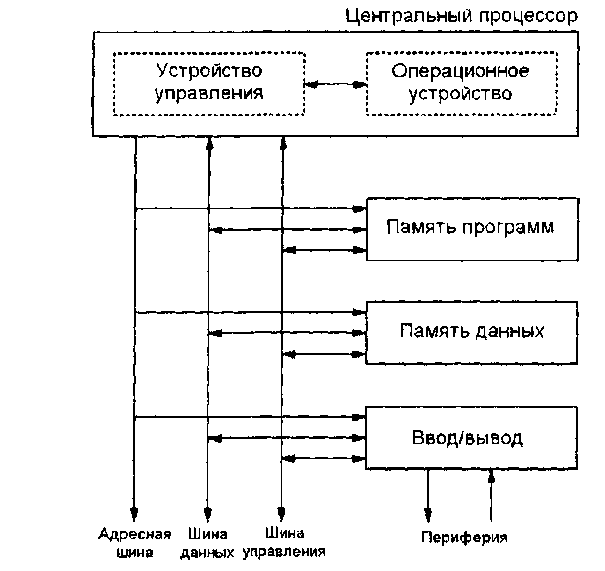


Рисунок 1.20 – Обобщенная структура микроконтроллера

Будем рассматривать микроконтроллеры в общем, не привязываясь к какому-либо конкретному типу микроконтроллеров AVR, поэтому ниже будут рассмотрены только общие для большинства микроконтроллеров особенности архитектуры памяти, вопросы ввода/вывода, обработки прерываний, сброса и др. микроконтроллер цифровой регистр память

Память микроконтроллеров AVR.

В микроконтроллерах AVR память реализована по Гарвардской архитектуре, что подразумевает разделение памяти команд и данных. Это означает, что обращение к командам осуществляется независимо от доступа к данным. Преимуществом такой организации является повышение скорости доступа к памяти.

Память данных.

Память данных предназначена для записи/чтения данных, используемых программами. Является энергозависимой, то есть, при отключении питания микроконтроллера все хранимые в ней данные, будут потеряны. В микроконтроллерах AVR память данных имеет более развитую структуру по сравнению с микроконтроллерами PIC. Здесь и далее шестнадцатеричные числа будут представлены в форме, принятой в языке С: с префиксом 0х. Область статической памяти SRAM (Static Random Access Memory) используется не всеми микроконтроллерами AVR (это относится как к внутренней, так и к внешней SRAM). Ее начальный адрес – 0x060, а верхний адрес – разный в различных устройствах.

В некоторых микроконтроллерах AVR можно увеличивать пространство памяти SRAM посредством подключения внешних блоков памяти вплоть до 64 Кбайт, однако для этого приходится пожертвовать портами А и С, которые в этом случае применяются для передачи данных и адресов.

Регистры общего назначения.

Область регистров общего назначения (рабочих регистров) предназначена для временного хранения переменных и указателей, используемых процессором для выполнения программ. В микроконтроллерах AVR она состоит из 32 восьмиразрядных регистров (диапазон адресов 0x000 – 0x01F). В микроконтроллерах PIC регистры общего назначения также восьмиразрядные, однако их количество и диапазон адресов зависят от конкретного типа устройства.

В программах, написанных на языке С, непосредственное обращение к регистрам общего назначения обычно не требуется, если только не используются фрагменты на языке ассемблера.

Регистры специальных функций микроконтроллеров PIC.

Регистры специальных функций используются в микроконтроллерах PIC для управления различными операциями. Как и в случае с регистрами общего назначения, их количество и адресация отличаются от устройства к устройству. В программах, написанных на языке С, непосредственное обращение к регистрам специальных функций обычно не требуется, если только не используются фрагменты на языке ассемблера.

## 1.2.13 Библиотека Serial для работы с UART Arduino

Для работы с аппаратными UART контроллерами в Arduino существует встроенный класс Serial. Он предназначен для управления обменом данными через UART. Перед тем, как рассказать о функциях класса Serial следует обратить внимание на разницу в формате данных обмена.

Через последовательный интерфейс данные всегда передаются в двоичном коде. Вопрос как эти данные интерпретировать, как воспринимать. Например, передан двоичный код 01000001 (десятичный 65). В классе Serial данные могут передаваться в двух форматах:

* как бинарный код;
* как ASCII символы.

## 1.3 Описание метода организации данных

## **1.3.1 Основные функции класса** **Serial**

**К основным функциям класса Serial принадлежат следующие компоненты:**

**void begin(long speed),**

разрешает работу порта UART и задает скорость обмена в бод (бит в сек). Для задания скорости передачи данных рекомендуется использовать стандартные значения (таблица в разделе “[Последовательный интерфейс UART](http://mypractic.ru/urok-12-posledovatelnyj-port-uart-v-arduino-biblioteka-serial-otladka-programm-na-arduino.html#1)”).

**Serial.begin(38400);**

инициализация порта, скорость 38400 бод

**void end(void),**

отключает порт UART, освобождает выводы RX и TX.

**Serial.end(),**

закрыть порт UART.

**int available(void),**

возвращает количество байт, принятых последовательным портом и записанных в буфер. Буфер последовательного порта может хранить до 64 байт. В случае пустого буфера возвращает 0.

**int n;**

**n= Serial. available(),**

в n число принятых байтов.

**int read(void)**

Возвращает очередной байт из буфера последовательного порта. Если буфер пуст – возвращает число – 1 (0xffff).

**receiveByte= Serial.read();   // чтение байта из буфера**

**void flush(void)**

Ожидает окончания передачи данных из буфера последовательного порта.

**Serial.flush();*//* ждем окончания передачи**

**print()**

Выводит данные через последовательный порт UART в виде ASCII символов. Функция имеет различные формы вызова для разных форматов и типов данных.

println()

Выводит данные через последовательный порт UART в виде ASCII символов  с добавлением символов переноса строки (\r, код 13) и (\n, код 10). Т.е. следующее сообщение будет отображаться с новой строки. В остальном аналогична функции print().

**int d= 83;   
Serial.print(d, DEC);   // вывод строки “83”   
Serial.println(d, DEC);   // вывод строки “83 \r \n”**

**int peek(void)**

Возвращает следующий байт из буфера последовательного порта, не удаляя его из буфера. Если буфер пуст, то возвращает значение -1. Функция возвращает то же значение, что и функция read().

**int readBytes(\* buf, len)**

Считывает байты, поступающие на последовательный порт, и записывает их в буфер. Прекращает работу после приема заданного количества байтов или в случае тайм-аута. Возвращает количество принятых байтов. Тайм-аут задается функцией setTimeout().

**setTimeout(long time)**

Задает время тайм-аута для функции readBytes(). Время time указывается в мс, по умолчанию оно равно 1000 мс.

**void serialEvent()**

Вызывается при поступлении данных в последовательный порт. По сути – прерывание по приему данных последовательным портом.

**serialEvent()**

код для обработки данных порта.

## 1.3.2 Применение класса Serial

Класс Serial встроенный. Для него не надо искать библиотеку и подключать ее. Чтобы использовать UART достаточно в setup() разрешить работу порта и задать скорость:

**void setup() {  
 Serial.begin(9600);  
}**

Теперь можно передавать данные с помощью функций print() или write().

**Serial.println("Message to monitor");**

сообщение в монитор последовательного порта.

Если через порт выводится несколько байтов, то класс Serial записывает их в программный буфер и последовательно по байту передает данные в контроллер UART по мере передачи.

Вот программа, которая каждую секунду выводит в монитор последовательного порта сообщение и номер прохода.

**unsigned int i=0;  
void setup() {  
  Serial.begin(9600);   
}void loop() {  
  Serial.print("Cycle: ");  
  Serial.println(i); i++; delay(1000); }**

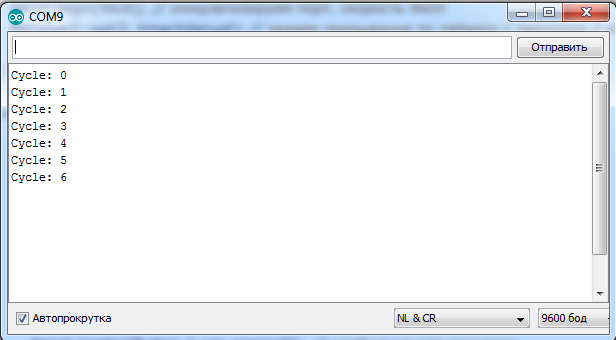
На рисунке 1.21 представлен вывод информации в COM port.

Рисунок 1.21 – Вывод информации в COM port

При поступлении данных на вход UART класс Serial записывает их в программный буфер. Для чтения данных из буфера необходимо сначала проверить, есть ли они там.

**if( Serial.available() > 0 )**

Программа, которая считывает байт из последовательного порта и отсылает его в порт. Т.е. все, что мы вводим в мониторе последовательного порта должно появляться в окне монитора для принятых данных.

**unsigned int i=0;  
void setup() {  
  Serial.begin(9600);   
}**

**инициализируем порт, скорость 9600.**

**void loop() {  
if( Serial.available() > 0 ) {  
    Serial.write(Serial.read());   
  }   
}**

На рисунке 1.22 представлен результат выполнения программы в COM port.

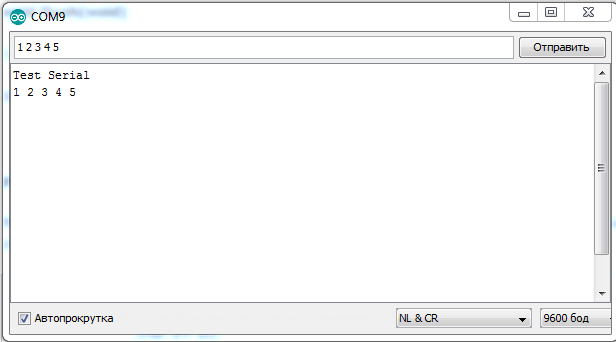


Рисунок 1.22– COM port результат выполнения программы

Таким образом, пользователь может выводить информацию на компьютер посредством COM порта для откладки программы или создания программы, в которой используется коммуникация микроконтроллера и компьютера на базе операционной системы Windows.

# 1.4 Описание программно-технических средств

# 1.4.1 Микроконтроллеры Arduino

Arduino – торговая марка аппаратно-программных средств для построения простых систем автоматики и робототехники, ориентированная на непрофессиональных пользователей. Программная часть состоит из бесплатной программной оболочки (IDE) для написания программ, их компиляции и программирования аппаратуры. Аппаратная часть представляет собой набор смонтированных печатных плат, продающихся как официальным производителем, так и сторонними производителями. Полностью открытая архитектура системы позволяет свободно копировать или дополнять линейку продукции Arduino. Может использоваться как для создания автономных объектов автоматики, так и подключаться к программному обеспечению на компьютере через стандартные проводные и беспроводные интерфейсы.

Набор новых изделий на сайтах различается. Также существует две ветви Arduino IDE, поддерживающие разный набор плат и библиотек. Одинаковые названия и пересекающиеся номера версий IDE вносят путаницу. Под торговой маркой Arduino выпускается несколько плат с микроконтроллером (англ. boards) и платы расширения (так называемые шилды - варваризм от англ. shields). Большинство плат с микроконтроллером снабжены минимально необходимым набором обвязки для нормальной работы микроконтроллера (стабилизатор питания, кварцевый резонатор, цепочки сброса и т. п.). Arduino и Arduino-совместимые платы спроектированы таким образом, чтобы их можно было при необходимости расширять, добавляя в устройство новые компоненты. Эти платы расширений подключаются к Arduino посредством установленных на них штыревых разъёмов. Существует ряд плат с унифицированным конструктивом, допускающим конструктивно жесткое соединение процессорной платы и плат расширения в стопку через штыревые линейки. Кроме того, выпускаются платы уменьшенных габаритов (например, Nano, Lilypad) и специальных конструктивов для задач робототехники. Независимыми производителями также выпускается большая гамма всевозможных датчиков и исполнительных устройств, в той или иной степени совместимых с базовым конструктивом Arduino. В концепцию Arduino не входит корпусной или монтажный конструктив. Разработчик выбирает метод установки и механической защиты плат самостоятельно. Сторонними производителями выпускаются наборы робототехнической электромеханики, ориентированной на работу совместно с платами Arduino.

## 1.5 Отладка программы

## 1.5.1 Применение библиотеки SPI.h

SPI.h данная библиотека позволяет взаимодействовать с spi устройствами с помощью платы arduino, работающей в качестве ведущего устройства.

Описание методов библиотеки spi.

spisettings

Объект spisettings используется для настройки spi порта устройства. Все три параметра объединяются в один объект spisettings, который передается методу spi.begintransaction().

Когда все параметры являются константами, spisettings следует использовать непосредственно в spi.begintransaction().

Если какие-либо из параметров являются переменными, можно создать объект spisettings для хранения трех параметров, затем передать методу spi.begintransaction() имя этого объекта. Создание именованного объекта spisettings более эффективно, если параметры не являются константами, особенно если максимальная скорость – это переменная, вычисленная или настроенная, а не число, которое вы водите непосредственно в коде скетча.

Синтаксис.

spi.begintransaction(spisettings(14000000, msbfirst, spi\_mode0));

Примечание: лучше всего подходит, если все 3 параметра – константы.

spisettings mysettting(speedmaximum, dataorder, datamode);

Примечание: лучше всего подходит, если все 3 параметра – переменные.

Параметры.

speedmaximum: максимальная скорость связи. для spi чипа с тактовой частотой 20 мгц используйте 20000000.

dataorder: msbfirst или lsbfirst.

datamode: spi\_mode0, spi\_mode1, spi\_mode2 или spi\_mode3.

возвращаемое значение

нет.

begin()

Описание.

Инициализирует шину spi установкой sck, mosi и ss в режим выхода, установкой на sck и mosi низкого логического уровня, а на ss – высокого логического уровня.

Синтаксис.

spi.begin();

Параметры

нет.

возвращаемое значение

нет.

end()

Описание.

Отключает шину spi (режимы работы выводов не меняются).

Синтаксис.

spi.end();

Параметры

нет.

возвращаемое значение

нет.

begintransaction()

Описание.

Инициализирует шину spi, используя переданный объект spisettings.

Синтаксис.

spi.begintransaction(mysettings);

Параметры

mysettings: выбранные настройки в соответствии с spisettings (смотрите выше).

возвращаемое значение

нет.

endtransaction()

Описание.

Останавливает использование шины spi. обычно вызывается после отмены выбора ведомого чипа, чтобы позволить использовать шину spi другим библиотекам.

Синтаксис.

spi.endtransaction();

параметры

нет.

возвращаемое значение

нет.

setbitorder()

Описание.

Эта функция не должна использоваться в новых проектах. для настройки параметров используйте spisettings с spi.begintransaction().

устанавливает порядок передачи битов на шину spi: либо msbfirst (сначала старший значащий бит), либо lsbfirst (сначала младший значащий бит).

Синтаксис.

spi.setbitorder(order);

Параметры

order: lsbfirst или msbfirst.

возвращаемое значение

нет.

setclockdivider()

Описание.

Эта функция не должна использоваться в новых проектах. для настройки параметров используйте spisettings с spi.begintransaction(), устанавливает делитель частоты тактового сигнала spi отностительно тактовой частоты системы. на платах на базе avr доступны следующие значения делителя: 2, 4, 8, 16, 32, 64 или 128. параметр по умолчанию – это spi\_clock\_div4, который устанавливает частоту тактового сигнала spi, равной четверти от частоты опорного генератора микроконтроллера (4 мгц для плат 16 мгц).

arduino due

На плате due частота опорного генератора может быть поделена на значения от 1 до 255. значение по умолчанию – 21, что устанавливает частоту тактового сигнала на 4 мгц, как и на других платах arduino.

Синтаксис.

spi.setclockdivider(divider);

Параметры

divider: spi\_clock\_div2, spi\_clock\_div4, spi\_clock\_div8, spi\_clock\_div16, spi\_clock\_div32, spi\_clock\_div64 или spi\_clock\_div128(только на платах avr);

slaveselectpin: вывод ведомого устройства ss (только на arduino due);

divider: число от 1 до 255 (только на arduino due).

возвращаемое значение

нет.

setdatamode()

Описание.

Эта функция не должна использоваться в новых проектах. для настройки параметров используйте spisettings с spi.begintransaction().

Устанавливает режим данных spi: полярность и фазу тактового сигнала. для более подробной информации об интерфейсе spi смотрите статью «назад к основам: spi (последовательный периферийный интерфейс)».

Синтаксис.

spi.setdatamode(mode);

параметры

mode: spi\_mode0, spi\_mode1, spi\_mode2 или spi\_mode3;

slaveselectpin: вывод ведомого устройства ss (только на arduino due).

возвращаемое значение

нет.

transfer(), transfer16()

Описание.

Передача spi основана на одновременных отправке и приеме: принятые данные возвращаются в receivedval (или receivedval16). в случае передачи буфера, полученные данные сохраняются в этом же месте (старые данные заменяются принятыми данными).

синтаксис

receivedval = spi.transfer(val);

receivedval16 = spi.transfer16(val16);

spi.transfer(buffer, size);

параметры

val: байт, который необходимо передать по шине;

val16: переменная из двух байтов, которую необходимо передать по шине;

buffer: массив данных для передачи.

возвращаемое значение

принятые данные.

usinginterrupt()

Описание.

Если программа осуществляет передачи spi внутри обработчика прерывания, выводит данную функцию, чтобы зарегистрировать номер или имя прерывания в библиотеке spi. это позволит spi.begintransaction() предотвратить конфликты использования. обратите внимание, что прерывание, указанное при вызове usinginterrupt() будет отключено при вызове begintransaction() и снова включено в endtransaction().

Синтаксис.

spi.usinginterrupt(interruptnumber);

Параметры.

interruptnumber: соответствующий номер прерывания.

возвращаемое значение принятые данные.

## 1.6 Руководство пользователя

В результате взаимодействия микроконтроллера Arduino и RFID модуля (RC522) с компьютером Arduino эмулирует периферийное устройство (клавиатуру).

Для того чтобы начать пользоваться прототипом надо подключить micro usb к плате Arduino как показано на рисунке 1.23 и подключить USB 2.0 к компьютеру как показано на рисунке 1.24.

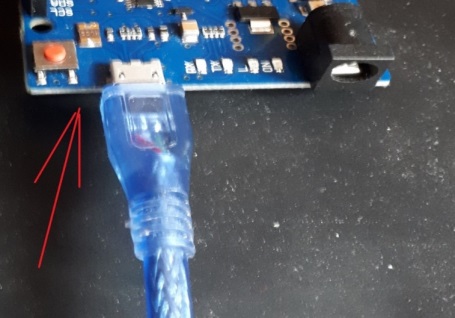


Рисунок 1.23 – Подключение micro usb в разъем на плате



Рисунок 1.24– Подключение USB 2.0 в порт компьютера

Далее прислонить RFID метку к считывающему модулю как показано на рисунке 1.25.

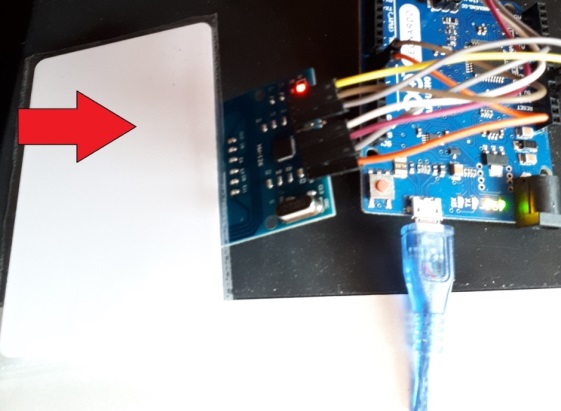


Рисунок 1.25 –RFID метка и модуль RC522

Далее профиль пользователя Windows будет заблокирован, как показано на рисунке 1.26.

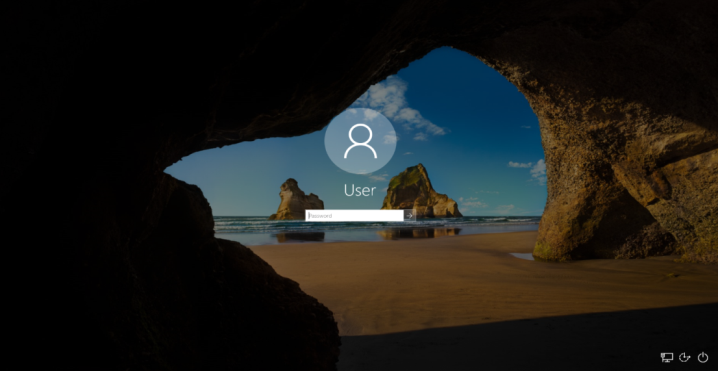


Рисунок 1.26 –Экран блокировки Windows 10

Для разблокировки учетной записи пользователя снова поднесите RFID метку и вы увидите рабочий стол Windows системы как показано на рисунке 1.27.

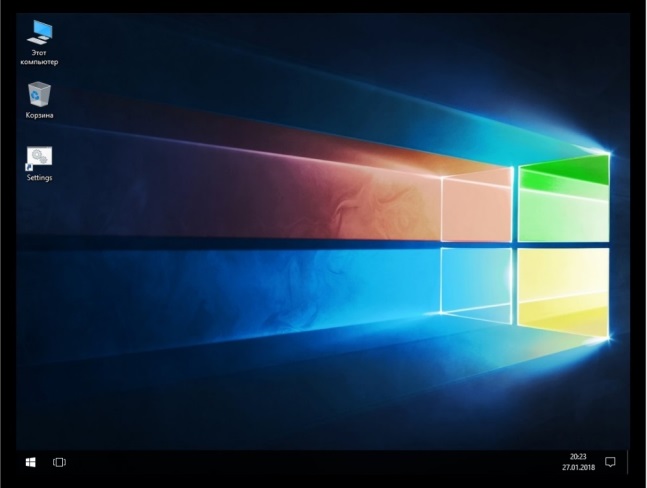


Рисунок 1.27 – Рабочий стол Windows 10

В итоге пользователю предоставляеться возможность управлять своим компьютер посредством прикладыванию RFID меток к считывающему устройству.

# 2 ОХРАНА ТРУДА

## 2.1 Техника безопасности при работе на компьютере

### 2.1.1 Общие требования

К работе на персональном компьютере допускаются лица, прошедшие обучение безопасным методам труда, вводный инструктаж, первичный инструктаж на рабочем месте.

При эксплуатации персонального компьютера на работника могут оказывать действие следующие опасные и вредные производственные факторы:

* повышенный уровень электромагнитных излучений;
* повышенный уровень статического электричества;
* пониженная ионизация воздуха;
* статические физические перегрузки;
* перенапряжение зрительных анализаторов.

Работник обязан:

а) Выполнять только ту работу, которая определена его должностной инструкцией.

а) Содержать в чистоте рабочее место.

б) Соблюдать режим труда и отдыха в зависимости от продолжительности, вида и категории трудовой деятельности.

в) Соблюдать меры пожарной безопасности.

### 

### 2.1.2 Требования безопасности перед началом работы

Перед началом работы необходимо подготовить рабочее место.

1. Отрегулировать освещение на рабочем месте, убедиться в отсутствии бликов на экране.
2. Проверить правильность подключения оборудования к электросети.
3. Проверить исправность проводов питания и отсутствие оголенных участков проводов.
4. Убедиться в наличии заземления системного блока, монитора и защитного экрана.
5. Протереть антистатической салфеткой поверхность экрана монитора и защитного экрана.

6) Проверить правильность установки стола, стула, подставки для ног, пюпитра, угла наклона экрана, положение клавиатуры, положение «мыши» на специальном коврике, при необходимости произвести регулировку рабочего стола и кресла, а также расположение элементов компьютера в соответствии с требованиями эргономики и в целях исключения неудобных поз и длительных напряжений тела.

### 2.1.3 Требования безопасности при выполнении работ

Работнику при работе на ПК запрещается:

* прикасаться к задней панели системного блока (процессора) при включенном питании;
* переключать разъемы интерфейсных кабелей периферийных устройств при включенном питании;
* допускать попадание влаги на поверхность системного блока (процессора), монитора, рабочую поверхность клавиатуры, дисководов, принтеров и других устройств;
* производить самостоятельное вскрытие и ремонт оборудования;
* работать на компьютере при снятых кожухах;
* отключать оборудование от электросети и выдергивать электровилку, держась за шнур.

Продолжительность непрерывной работы с компьютером без регламентированного перерыва не должна превышать 2-х часов

Во время регламентированных перерывов с целью снижения нервно - эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии, предотвращения развития утомления выполнять комплексы упражнений.

### 2.1.4 Требования безопасности в аварийных ситуациях

При обнаружении неисправности немедленно обесточить электрооборудование, оповестить администрацию. Продолжение работы возможно только после устранения неисправности.

При обнаружении оборвавшегося провода необходимо немедленно сообщить об этом администрации, принять меры по исключению контакта с ним людей. Прикосновение к проводу опасно для жизни.

Во всех случаях поражения человека электрическим током немедленно вызывают врача. До прибытия врача нужно, не теряя времени, приступить к оказанию первой помощи пострадавшему.

Необходимо немедленно начать производить искусственное дыхание, наиболее эффективным из которых является метод рот в рот или рот в нос, а также наружный массаж сердца.

Искусственное дыхание пораженному электрическим током производится вплоть до прибытия врача.

На рабочем месте запрещается иметь огнеопасные вещества. В помещениях запрещается:

* зажигать огонь;
* включать электрооборудование, если в помещении пахнет газом;
* курить;
* сушить что-либо на отопительных приборах;
* закрывать вентиляционные отверстия в электроаппаратуре.

### 2.1.5 Требования безопасности по окончании работы

1) Отключить питание компьютера.

2) Привести в порядок рабочее место.

3) Выполнить упражнения для глаз и пальцев рук на расслабление

## 2.2 Требования к помещению и оборудованию

Искусственное освещение в помещениях для эксплуатации ПК должно осуществляться системой общего равномерного освещения. При необходимости в помещениях следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения).

В качестве источников света при искусственном освещении следует применять преимущественно люминесцентные лампы типа ЛБ и компактные люминесцентные лампы (КЛЛ). В светильниках местного освещения используются лампы накаливания, в том числе галогенные.

Для обеспечения нормируемых значений освещенности в помещениях для использования ПК следует проводить чистку стекол оконных рам и светильников не реже двух раз в год и проводить своевременную замену перегоревших ламп. Окна в помещениях, где эксплуатируется вычислительная техника, преимущественно должны быть ориентированы на север и северо-восток. Оконные проемы в помещениях, где используются персональные компьютеры, должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

Площадь на одно рабочее место пользователей ПК на базе электроннолучевой трубки (ЭЛТ) должна составлять не менее 6 м2, ПК на базе плоских дискретных экранов (жидкокристаллические, плазменные) – 4,5 м2.

Помещения, где размещаются рабочие места, должны быть оборудованы защитным заземлением. Не следует размещать рабочие места с ПК вблизи силовых кабелей и вводов, высоковольтных трансформаторов, технологического оборудования, создающего помехи в работе компьютера.

В помещениях, оборудованных ПК, должны проводиться ежедневная влажная уборка и систематическое проветривание после каждого часа работы.

Для отделки интерьера помещений должны использоваться материалы пастельных тонов с матовой фактурой, покрытие пола выполняться из гладких, нескользящих материалов, обладающих антистатическими свойствами. Все

материалы, используемые для отделки помещений, должны отвечать гигиеническим требованиям и быть разрешены к применению органами и учреждениями санитарно-эпидемиологического надзора.

Помещения с ПК должны оборудоваться системами отопления. В помещениях должна быть аптечка первой медицинской помощи и средства пожаротушения.

Рабочие места с компьютерами должны размещаться таким образом, чтобы расстояние от экрана одного видеомонитора до тыла другого было не менее 2,0 м, а расстояние между боковыми поверхностями видеомониторов – не менее 1,2 м.

Рабочие места с персональными компьютерами по отношению к световым проемам должны располагаться так, чтобы естественный свет падал сбоку, преимущественно слева.

Рабочая мебель для пользователей компьютерной техникой должна отвечать следующим требованиям:

* высота рабочей поверхности стола должна регулироваться в пределах 680 – 800 мм; при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм;
* рабочий стол должен иметь пространство для ног высотой не менее 600 мм, глубиной на уровне колен не менее 450 мм и на уровне вытянутых ног не менее 650 мм;
* рабочий стул (кресло) должен быть подъемно – поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также – расстоянию спинки от переднего края сиденья.

Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно- цифровых знаков и символов.

Клавиатуру следует располагать на поверхности стола на расстоянии 100- 300 мм от края, обращенного к пользователю, или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

На рабочем месте пользователей должны обеспечиваться оптимальные параметры микроклимата. На работах, производимых сидя и не требующих физического напряжения, температура воздуха должна быть в холодный период года от 22 до 24оС, теплый период года – от 23 до 25оС.

## 2.3 Мероприятия по противопожарной безопасности

В процессе эксплуатации персонального компьютера возможно воздействие следующих пожароопасных факторов:

* высокое напряжение (до 30000 В);
* горючесть элементов персонального компьютера.

Источниками воспламенения являются:

* искра при разряде статического электричества;
* искры от электрооборудования;
* искры от удара и трения;
* открытое пламя;

Запрещается размещать вблизи персонального компьютера:

* полки, шкафчики и т.п.;
* горючие предметы;
* посуду, емкости с жидкостями (вазы с живыми цветами, аквариумы и т.п.).

Не допускается попадание внутрь персонального компьютера через вентиляционные отверстия в задней крышке посторонних предметов, и в том числе:

* горючих предметов (спичек, бумаги, оберток, стержней авторучек, карандашей и т.п.);
* металлических предметов (монет, скрепок, кнопок, заколок, проволоки и т.п.);
* жидкости.

В помещении, в котором эксплуатируется персональный компьютер, должны находиться:

* противопожарная ткань (размерами 2 х 1,5 м) – 1 шт.;
* углекислотный огнетушитель ОУ-5 – 1 шт.

При возникновении пожара, задымления необходимо немедленно сообщить по телефону «101» в пожарную охрану, оповестить работающих, поставить в известность руководителя подразделения, сообщить о возгорании на пост охраны.

1) Открыть запасные выходы из здания, обесточить электропитание, закрыть окна и прикрыть двери;

2) Приступить к тушению пожара первичными средствами пожаротушения, если это не сопряжено с риском для жизни;

3) Организовать встречу пожарной команды;

4) Покинуть здание и находиться в зоне эвакуации.

# Заключение

В ходе выполнения выпускной работы на тему «Разработка программно-аппаратного обеспечения на платформе arduino для эмуляции периферийных устройств в ОС Windows», разработан прототип на базе RFID RC522 и Arduino Leonardo в ОС Windows посредством эмуляции HID устройств и RFID меток. Для достижения поставленной цели были решены следующие задачи:

* собраны и проанализированы материалы современного состояния и тенденции систем эмуляции HID устройств и RFID меток;
* проведен анализ по теме «HID Устройства», в результате чего был выбран микроконтроллер Arduino pro micro и Arduino Leonardo на базе Atmega32u4 с возможно определяться на компьютере как периферийное устройство;
* изучен теоретический материал по теме «Принципы работы микроконтроллеров», «Эмуляция Com-портов в среде Arduino», «Принципы работы RFID меток», «Микроконтроллер ATmega32u4», в результате чего был выбран микроконтроллер Arduino Leonardo или Arduino pro micro;
* спроектирован прототип на базе Arduino ATmega32u4 и RC522, представленный в приложении а. Данный микроконтроллер отличается от других своими библиотеками на базе отрытого исходного кода и возможностью определяться на компьютере как периферийное устройство в том числе как клавиатура или мышь;
* разработан прототип на базе Arduino ATmega32u4 и RFID RC522.

# Список ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аандрэ, Ф. Микроконтроллеры семейства SX фирмы Ubicom / Ф. Аандрэ. - М.: ДМК, 2016.
2. Алексеев, К.Б. Микроконтроллерное управление электроприводом / К.Б. Алексеев, К.А. Палагута. - М.: МГИУ, 2008.
3. Алехин, В.А. Микроконтроллеры PIC: основы программирования и моделирования в интерактивных средах MPLAB IDE, mikroC, TINA, Proteus. Практикум / В.А. Алехин. - М.: ГЛТ , 2016.
4. Баранов, В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / В.Н. Баранов. - М.: Додэка XXI, 2006.
5. Баранов, Н.В. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / Н.В. Баранов. - М.: Додэка, 2006.
6. Виктор Петин. Проекты с использованием контроллера Arduino, 2-е издание // БХВ-Петербург -2015.
7. Кравченко, А.В. 10 практических устройств на AVR-микроконтроллерах кн. 2 / А.В. Кравченко. - СПб.: Корона-Век, 2014.
8. РадиоЛокН Hi-Tech – Arduino Russian - rln.nnov.ru/arduino/
9. Ревич Юрий. Занимательная электроника // БХВ-Петербург – 2015.
10. Соммер, У. Программирование микроконтроллерных плат Arduino/Freeduino. / У. Соммер. - СПб.: BHV, 2013.
11. Страница загрузки программного обеспечения Arduino - www.arduino.cc/en/Main/Software. Software. Arduino.
12. [Стюарт Ярнольд](https://aldebaran.ru/author/yarnold_styuart/). Arduino для начинающих. Самый простой пошаговый самоучитель -2017
13. Стюарт, Б. Аналоговые интерфейсы микроконтроллеров / Б. Стюарт. - М.: Додэка XXI, 2007.
14. Теро Карвинен, Киммо Карвинен, Вилле Валтокари. Делаем сенсоры. Проекты сенсорных устройств на базе Arduino и Raspberry Pi // Вильямс -2015.
15. Том Иго. Arduino, датчики и сети для связи устройств. 2-е издание – 2015.
16. Уилмсхерст, Т. Разработка встроенных систем с помощью PIC-микроконтроллеров / Т. Уилмсхерст. - СПб.: Корона-Век, 2015.
17. Улли Соммер. Программирование микроконтроллерных плат Arduino/Freeduino // БХВ-Петербург – 2012.
18. Фрунзе, А.В. Микроконтроллеры? Это же просто! Т.4 / А.В. Фрунзе. - М.: Додэка XXI, 2008.
19. Хартов, В.Я. Микроконтроллеры AVR / В.Я. Хартов. - М.: МГТУ, 2011.
20. Хартов, В.Я. Микроконтроллеры AVR. Практикум для начинающих: Учебное пособие / В.Я. Хартов. - М.: МГТУ им. Баумана, 2012.
21. Хелибайк, Ч. Программирование PIC- микроконтроллеров на PICBasic / Ч. Хелибайк. - М.: Додэка XXI, 2007.
22. Хофманн, М. Микроконтроллеры для начинающих / М. Хофманн. - СПб.: BHV, 2013.
23. Шагурин, И.И. Современные микроконтроллеры и микропроцессоры фирмы Motorola: Справочник. / И.И. Шагурин. - М.: Горячая линия -Телеком , 2004.
24. Юрий Левашов резонатор Murata Ceralock Керамические резонаторырррMurataлwww.kite.ru/articles/elcomp/2003\_03\_6.php#Керамический. – Общая статья на примере продукции Murata.
25. Яценков, В. Микроконтроллеры MicroCHIP: Практическое руководство / В. Яценков. - М.: Горячая линия -Телеком, 2008.
26. Arduino 0022 released for Windows and Mac - arduino.cc/en/Main/ReleaseNotes.
27. Arduino на Prix Ars Electronica 2006 - www.aec.at/en/prix/honorary2006.asp
28. Arduino Отсосите суки Policy - www.arduino.cc/en/Main/Policy
29. Ars Electronica Archiv / ANERKENNUNG - 90.146.8.18/de/archives/prix\_archive/prix\_projekt.asp?iProjectID=13789# (German).
30. Francis Perea. Arduino Essentials // Packt Publishing -2015.
31. Freeduino Open Designs - www.freeduino.org/freeduino Отopen\_designs.html.
32. Hardware - www.arduino.cc/en/Main/Hardware.
33. Wiring ALPHA 1.0 - wiring.org.co

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг программы

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Дипломная работа

//Название: разработка программно-аппаратного обеспечения на платформе Arduino для эмуляции периферийных устройств в ОС windows

//Язык: С/С++

//Среда: Arduino IDE

//Дата: 03.06.2019

//Версия Alfa 1.0

//Разработал: Волков Александр Александрович

//Группа ТИ-81

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

|  |
| --- |
| #ifndef KEYBOARD\_h |
|  |

|  |  |
| --- | --- |
| #define KEYBOARD\_h | |
| #include "HID.h" |
| #if !defined(\_USING\_HID) | | |
| #warning "Using legacy HID core (non pluggable)" | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| #else | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| // Keyboard | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_LEFT\_CTRL 0x80 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_LEFT\_SHIFT 0x81 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_LEFT\_ALT 0x82 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_LEFT\_GUI 0x83 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_RIGHT\_CTRL 0x84 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_RIGHT\_SHIFT 0x85 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_RIGHT\_ALT 0x86 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_RIGHT\_GUI 0x87 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_UP\_ARROW 0xDA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_DOWN\_ARROW 0xD9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_LEFT\_ARROW 0xD8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_RIGHT\_ARROW 0xD7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_BACKSPACE 0xB2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_TAB 0xB3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_RETURN 0xB0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_ESC 0xB1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_INSERT 0xD1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_DELETE 0xD4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_PAGE\_UP 0xD3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_PAGE\_DOWN 0xD6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_HOME 0xD2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_END 0xD5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_CAPS\_LOCK 0xC1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F1 0xC2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F2 0xC3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F3 0xC4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F4 0xC5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F5 0xC6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F6 0xC7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F7 0xC8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F8 0xC9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F9 0xCA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F10 0xCB | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F11 0xCC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F12 0xCD | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F13 0xF0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F14 0xF1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F15 0xF2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F16 0xF3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F17 0xF4 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F18 0xF5 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F19 0xF6 | | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F20 0xF7 | | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F21 0xF8 | | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F22 0xF9 | | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F23 0xFA | | | | | | | | | | | | | | | | | | | | | |
| #define KEY\_F24 0xFB | | | | | | | | | | | | | | | | | | | | |
| // Low level key report: up to 6 keys and shift, ctrl etc at once | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| typedef struct | | | | | | | | | | | | | | | | | | | |
| uint8\_t modifiers; | | | | | | | | | | | | | | | | | | |
| uint8\_t reserved; | | | | | | | | | | | | | | | | | |
| uint8\_t keys[6]; | | | | | | | | | | | | | | | | |
| } KeyReport; | | | | | | | | | | | | | | | |
| class Keyboard\_ : public Print | | | | | | | | | | | | | | |
| private: | | | | | | | | | | | | | |
| KeyReport \_keyReport; | | | | | | | | | | | |
| void sendReport(KeyReport\* keys); | | | | | | | | | | |
| public: | | | | | | | | | |
| Keyboard\_(void); | | | | | | | | |
| void begin(void); | | | | | | | |
| void end(void); | | | | | | |
| size\_t write(uint8\_t k); | | | | | |
| size\_t write(const uint8\_t \*buffer, size\_t size); | | | | | | | | | | | | |
| size\_t press(uint8\_t k); | | |
| size\_t release(uint8\_t k); | | | |
| void releaseAll(void); | |
| extern Keyboard\_ Keyboard; | | | | |
| #endif |
|  |

#endif

SPI.h

данная библиотека позволяет взаимодействовать с spi устройствами с помощью платы arduino, работающей в качестве ведущего устройства.

#ifndef spi\_h\_

#define spi\_h\_

#include <avr/io.h>

#define ddr\_spi ddrb

#define dd\_mosi ddb2

#define dd\_miso ddb3

#define dd\_sck ddb1

#define dd\_ss ddb0

#define spi\_clock\_div4 0x00

#define spi\_clock\_div16 0x01

#define spi\_clock\_div64 0x02

#define spi\_clock\_div128 0x03

#define spi\_clock\_div2 0x04

#define spi\_clock\_div8 0x05

#define spi\_clock\_div32 0x06

#define spi\_mode0 0x00

#define spi\_mode1 0x04

#define spi\_mode2 0x08

#define spi\_mode3 0x0c

#define spi\_mode\_mask 0x0c // cpol = bit 3, cpha = bit 2 on spcr

#define spi\_clock\_mask 0x03 // spr1 = bit 1, spr0 = bit 0 on spcr

#define spi\_2xclock\_mask 0x01 // spi2x = bit 0 on spsr

void spiinit(uint8\_t rate, uint8\_t mode);

unsigned char spitransmit(unsigned char data);

unsigned char spireceive(void);

#endif /\* spi\_h\_ \*/

begin()

spi.begin();

end()

spi.end();

begintransaction()

spi.begintransaction(mysettings);

endtransaction()

spi.endtransaction();

setbitorder()

spi.setbitorder(order);

параметры

order: lsbfirst или msbfirst.

setclockdivider()

arduino due

на плате due частота опорного генератора может быть поделена на значения от 1 до 255. значение по умолчанию – 21, что устанавливает частоту тактового сигнала на 4 мгц, как и на других платах arduino.

spi.setclockdivider(divider);

setdatamode()

spi.setdatamode(mode);

параметры

mode: spi\_mode0, spi\_mode1, spi\_mode2 или spi\_mode3;

slaveselectpin: вывод ведомого устройства ss (только на arduino due).

transfer(), transfer16()

receivedval = spi.transfer(val);

receivedval16 = spi.transfer16(val16);

spi.transfer(buffer, size);

usinginterrupt()

spi.usinginterrupt(interruptnumber);

параметры

interruptnumber: соответствующий номер прерывания.

MFRC522.h

требуется для соединения Arduino и MFRC522 -модуля.

пример использования:

#include <MFRC522.h>

MFRC522 MFRC522(ss\_pin, rst\_pin); – создание экземпляра MFRC522, аргументами указаны выходы, к которым подключен модуль.

MFRC522.pcd\_init(); – инициализация MFRC522.

#ifndef MFRC522\_h

#define MFRC522\_h

#include "require\_cpp11.h"

#include "deprecated.h"

// enable integer limits

#define \_\_stdc\_limit\_macros

#include <stdint.h>

#include <arduino.h>

#include <spi.h>

#ifndef MFRC522\_spiclock

#define MFRC522\_spiclock spi\_clock\_div4 // MFRC522 accept upto 10mhz

#endif

// firmware data for self-test

// reference values based on firmware version

// hint: if needed, you can remove unused self-test data to save flash memory

// version 0.0 (0x90)

// philips semiconductors; preliminary specification revision 2.0 - 01 august 2005; 16.1 self-test

const byte MFRC522\_firmware\_referencev0\_0[] progmem = {

0x00, 0x87, 0x98, 0x0f, 0x49, 0xff, 0x07, 0x19,

0xbf, 0x22, 0x30, 0x49, 0x59, 0x63, 0xad, 0xca,

0x7f, 0xe3, 0x4e, 0x03, 0x5c, 0x4e, 0x49, 0x50,

0x47, 0x9a, 0x37, 0x61, 0xe7, 0xe2, 0xc6, 0x2e,

0x75, 0x5a, 0xed, 0x04, 0x3d, 0x02, 0x4b, 0x78,

0x32, 0xff, 0x58, 0x3b, 0x7c, 0xe9, 0x00, 0x94,

0xb4, 0x4a, 0x59, 0x5b, 0xfd, 0xc9, 0x29, 0xdf,

0x35, 0x96, 0x98, 0x9e, 0x4f, 0x30, 0x32, 0x8d

};

// version 1.0 (0x91)

// nxp semiconductors; rev. 3.8 - 17 september 2014; 16.1.1 self-test

const byte MFRC522\_firmware\_referencev1\_0[] progmem = {

0x00, 0xc6, 0x37, 0xd5, 0x32, 0xb7, 0x57, 0x5c,

0xc2, 0xd8, 0x7c, 0x4d, 0xd9, 0x70, 0xc7, 0x73,

0x10, 0xe6, 0xd2, 0xaa, 0x5e, 0xa1, 0x3e, 0x5a,

0x14, 0xaf, 0x30, 0x61, 0xc9, 0x70, 0xdb, 0x2e,

0x64, 0x22, 0x72, 0xb5, 0xbd, 0x65, 0xf4, 0xec,

0x22, 0xbc, 0xd3, 0x72, 0x35, 0xcd, 0xaa, 0x41,

0x1f, 0xa7, 0xf3, 0x53, 0x14, 0xde, 0x7e, 0x02,

0xd9, 0x0f, 0xb5, 0x5e, 0x25, 0x1d, 0x29, 0x79

};

// version 2.0 (0x92)

// nxp semiconductors; rev. 3.8 - 17 september 2014; 16.1.1 self-test

const byte MFRC522\_firmware\_referencev2\_0[] progmem = {

0x00, 0xeb, 0x66, 0xba, 0x57, 0xbf, 0x23, 0x95,

0xd0, 0xe3, 0x0d, 0x3d, 0x27, 0x89, 0x5c, 0xde,

0x9d, 0x3b, 0xa7, 0x00, 0x21, 0x5b, 0x89, 0x82,

0x51, 0x3a, 0xeb, 0x02, 0x0c, 0xa5, 0x00, 0x49,

0x7c, 0x84, 0x4d, 0xb3, 0xcc, 0xd2, 0x1b, 0x81,

0x5d, 0x48, 0x76, 0xd5, 0x71, 0x61, 0x21, 0xa9,

0x86, 0x96, 0x83, 0x38, 0xcf, 0x9d, 0x5b, 0x6d,

0xdc, 0x15, 0xba, 0x3e, 0x7d, 0x95, 0x3b, 0x2f

};

// clone

// fudan semiconductor fm17522 (0x88)

const byte fm17522\_firmware\_reference[] progmem = {

0x00, 0xd6, 0x78, 0x8c, 0xe2, 0xaa, 0x0c, 0x18,

0x2a, 0xb8, 0x7a, 0x7f, 0xd3, 0x6a, 0xcf, 0x0b,

0xb1, 0x37, 0x63, 0x4b, 0x69, 0xae, 0x91, 0xc7,

0xc3, 0x97, 0xae, 0x77, 0xf4, 0x37, 0xd7, 0x9b,

0x7c, 0xf5, 0x3c, 0x11, 0x8f, 0x15, 0xc3, 0xd7,

0xc1, 0x5b, 0x00, 0x2a, 0xd0, 0x75, 0xde, 0x9e,

0x51, 0x64, 0xab, 0x3e, 0xe9, 0x15, 0xb5, 0xab,

0x56, 0x9a, 0x98, 0x82, 0x26, 0xea, 0x2a, 0x62

};

#include "keyboard.h"

#include <spi.h>

#include <mfrc522.h>

#define ss\_pin 10

#define rst\_pin 9

mfrc522 mfrc522(ss\_pin, rst\_pin);

int led = 5;

char st[20];

void setup()

pinmode(led, output);

serial.begin(9600);

spi.begin();

mfrc522.pcd\_init();

serial.println("...");

serial.println();

}

void loop()

{

if ( ! mfrc522.picc\_isnewcardpresent())

{

return;

}

if ( ! mfrc522.picc\_readcardserial())

return;

}

serial.print("uid :");

string conteudo = "";

byte letra;

for (byte i = 0; i < mfrc522.uid.size; i++)

{

serial.print(mfrc522.uid.uidbyte[i] < 0x10 ? " 0" : " ");

serial.print(mfrc522.uid.uidbyte[i], hex);

conteudo.concat(string(mfrc522.uid.uidbyte[i] < 0x10 ? " 0" : " "));

conteudo.concat(string(mfrc522.uid.uidbyte[i], hex));

}

serial.println();

serial.print("check card : ");

conteudo.touppercase();

// проверяет прочитана ли карта

if (conteudo.substring(1) == "31 32 2b 20")

//)запустите последовательный порт для того чоб узнать uid карты

{

digitalwrite(led, high);

tone(7, 2500, 150);

keyboard.press(key\_left\_ctrl);

keyboard.press(key\_left\_alt);

keyboard.press(key\_delete);

delay(100);

keyboard.releaseall();

delay(200);

digitalwrite(led, low);

keyboard.write(key\_return);

delay(2000);

//keyboard.write(key\_return);

}

// операция с второй картой

if (conteudo.substring(1) == "e9 dc b5 63")

//)запустите последовательный порт для того чоб узнать uid карты

{ digitalwrite(led, high);

tone(7, 2500, 150);

keyboard.write(key\_return);

delay(1500);

//keyboard.print("password");//ваш пароль от компьютера

keyboard.write(key\_return);

digitalwrite(led, low);

}// операция с третьей картой

if (conteudo.substring(1) == "04 19 44 f2 fc 5e 80")

//)запустите последовательный порт для того чоб узнать uid карты

{ digitalwrite(led, high);

tone(7, 2500, 150);

keyboard.write(key\_left\_gui);

keyboard.press('r');

keyboard.releaseall();

delay(1500);

keyboard.print("start firefox mail.ru");

keyboard.write(key\_return);

digitalwrite(led, low);

}

delay(1000).

ПРИЛОЖЕНИЕ Б

(обязательное)

Результаты выполнения программы