

# Азбука халтурщика-ARMатурщика *разработка встраиваемых систем*

основы бытовой автоматики,  
систем управления и сбора данных

- © ruOpenWrt
- © HackSpace «Чебураторный завод»
- © Консорциум хоббитов России
- © Bill Collis (Часть I)

# Оглавление

Введение	15
<b>I Введение в практическую электронику</b>	<b>16</b>
An Introduction to Practical Electronics, Microcontrollers and Software Design © Bill Collis . . . . .	17
<b>1 1 Введение в практическую электронику</b>	<b>18</b>
1.1 Ваше обучение по специальности «Технология» . . . . .	19
1.1.1 Цели обучения технологиям Ново-Зеландской программы . . . . .	19
1.2 Ключевые компетенции Ново-Зеландской программы . . . . .	21
<b>2 2 Вводная электронная схема</b>	<b>23</b>
<b>3 3 Вводное конструирование печатной платы</b>	<b>24</b>
<b>4 4 Пайка, припой и паяльники</b>	<b>25</b>

5	5 Введение в теорию электроники	49	26
6	6 Введение в электронику микроконтроллера	63	27
7	7 Входные цепи микроконтроллера	91	28
8	8 Обзор программирования	104	29
9	9 Введение в поток выполнения программы	112	30
10	10 Вводное программирование: использование подпрограмм	126	31
11	11 Вводное программирование: Использование переменных	134	32
12	12 Основные дисплеи	161	33
13	13 Проект портативного аудиоусилителя на TDA2822M	174	34
14	14 Основы логического программирования	187	35
15	15 Разработка алгоритма: Система сигнализации	202	36
16	16 Основы теории цепей постоянного тока	215	37
17	17 Основы планирования проекта	236	38
18	18 Пример дизайна системы: Таймер клеевого пистолета	268	39

19	19 Основные интерфейсы и их программирование	273	40
20	20 Основы интерфейса аналого-цифрового преобразования	295	41
21	21 Основы проектирования системы	314	42
22	22 Основы проектирования системы: Тайм-трекер	317	43
23	23 Основы вычислений времени	330	44
24	24 Основы строковых переменных	340	45
25	25 Силовые интерфейсы	353	46
26	26 Теория источников питания	370	47
27	27 Типичные вопросы тестирования 2011/12/13 годов	395	48
28	28 Расширенное программирование: Массивы	397	49
29	29 Подтягивающие резисторы AVR	402	50
30	30 Дополнительноподключение клавиатуры	403	51
31	31 Тонкости циклов Do-Loop & While-Wend	417	52
32	32 Подключение двигателя постоянного тока	423	53

33 33	Пример расширенной системы: Будильник	452	54
34 34	Резистивный сенсорный экран	468	55
35 35	Пример проектирования системы: Регулятор температуры	475	56
36 36	Расширенное программирование: Машины состояний	478	57
37 37	Переработанный проект будильника	501	58
38 38	Студенческий проект: Расширенный оконный контроллер	514	59
39 39	Альтернативные техники кодирования машин состояния	524	60
40 40	Сложно: последовательная связь	526	61
41 41	Цифровой радиоканал	597	62
42 42	Введение в I2C	617	63
43 43	Студенческий проект: Таймер полива теплицы	631	64
44 44	Проект Велосипедного аудиоусилителя	642	65
45 45	Графические LCD	648	66
46 46	Проект Отслеживания температуры GLCD	660	67

47 47 Прерывания 672	68
48 48 Таймеры/Счётчики 692	69
49 49 Проект скроллинга графического LED дисплея: массивы и таймеры 698	70
50 50 Проект медицинского прибора: реализация таймера 709	71
51 Проект часов на 7-сегментном индикаторе реализация на сдвоенном таймере	72
52 52 ИС драйвера дисплея MAX 7219/7221 739	73
53 53 Подключение через мобильную связь: ADH8066 744	74
54 54 Передача данных через Internet 778	75
55 55 Задание: математика в реальном мире 816	76
56 56 Цветной графический LCD на основе SSD1928 825	77
57 57 Светофор: помощь и решение 865	78
58 58 Компьютерное программирование: низкоуровневые детали 869	79
59 59 USB-программатор: USBASP 876	80

60 60 Программатор USBTinyISP 877	81
61 61 Программирование на Си и AVR 881	82
62 62 Объектно-Оrientированное Программирование (ООП) на $C_{+}^{+}$ и AVR 929	83
63 63 Современные (2014) отладочные платы на AVR 953	84
64 64 Eagle: создание собственной библиотеки 970	85
65 65 Практические методы 979	86
66 66 ЧПУ 990	87
67 67 Индекс 1008	88
 II Основы электроники	 89
68 Линейные схемы на пассивных элементах, основы электротехники	91
69 Симуляция и расчет схем в ngSPICE	92
70 KiCAD	93
70.1 Отрисовка схем в KiCAD	93
70.2 Библиотеки элементов	93
70.3 Передача схемы в ngSPICE	93

<b>71 Простейшие полупроводниковые элементы</b>	<b>94</b>
71.1 Оптоэлектроника . . . . .	94
71.2 Схемы на биполярных транзисторах . . . . .	94
71.3 Схемы на полевых транзисторах . . . . .	94
<b>72 Операционные усилители</b>	<b>95</b>
<b>73 Источники питания</b>	<b>96</b>
73.1 Батарейное питание . . . . .	96
73.2 Линейные стабилизаторы . . . . .	96
73.3 Импульсные преобразователи на ШИМ-контроллерах . . . . .	96
73.4 Цепи защиты и гашения кондуктивных помех . . . . .	96
<b>74 Цифровая электроника</b>	<b>97</b>
<b>75 Компьютерные интерфейсы</b>	<b>98</b>
75.1 Поколение 90х: COM, LPT, ISA . . . . .	99
75.1.1 Резервный программатор AVR “пять проводков” . . . . .	99
75.2 Сеть CAN . . . . .	99
75.3 Интерфейсные модули USB . . . . .	99
75.3.1 Универсальный высокоскоростной конвертер FTDI FT232RL . . . . .	99
75.3.2 JTAG-адаптер . . . . .	99
75.3.3 Отладочный модуль CAN . . . . .	99
75.4 Интерфейсные модули Ethernet . . . . .	99
<b>76 ПЛИС</b>	<b>100</b>




<b>77 Датчики</b>	<b>101</b>
<b>78 Электропривод и исполнительные устройства</b>	<b>102</b>
 <b>III Основы конструирования РЭС</b>	 <b>103</b>
<b>79 Пакеты моделирования на основе OpenFOAM</b>	<b>104</b>
<b>80 Обеспечение теплового режима</b>	<b>105</b>
<b>81 Электромагнитная совместимость</b>	<b>106</b>
81.1 Кондуктивные помехи	106
81.2 Компонентные модели и оптимизация кабельной сети	106
 <b>IV Технология РЭС</b>	 <b>107</b>
<b>82 Инструменты и оборудование</b>	<b>108</b>
82.1 JTAG-адаптер	108
82.2 Отладочные платы	108
82.2.1 Arduino /Atmel Mega AVR8/	109
82.2.2 Cortex-Mx	109
82.2.3 CubieBoard /Cortex-A8 AllWinner A10/	110
82.2.4 Raspberry Pi /ARM11 BCM3032/	110
82.2.5 BlackSwift /MIPS/	110

82.2.6 VoCore /MIPS/ . . . . .	110
82.3 Монтажный инструмент . . . . .	110
82.4 Измерительное оборудование . . . . .	110
82.4.1 Тестер . . . . .	110
82.4.2 Осциллограф . . . . .	110
82.4.3 Логический анализатор . . . . .	110
82.4.4 Генератор сигналов . . . . .	110
82.4.5 Рыльцемер . . . . .	110
82.5 Электроинструмент . . . . .	111
82.5.1 Дрель . . . . .	111
<b>83 Трассировка плат и подготовка производства в KiCAD</b>	<b>112</b>
83.1 Технология ЛУТ (Лазерный УТюг) . . . . .	112
83.2 Технология фоторезиста . . . . .	112
83.3 Формат Gerber и подготовка промышленного производства . . . . .	112
<b>84 FreeCAD</b>	<b>113</b>
84.1 Установка под Windows . . . . .	115
84.2 Чертеж . . . . .	117
84.3 Эскиз . . . . .	117
84.4 Деталь . . . . .	117
84.5 Сборка . . . . .	117
84.6 Автогенерация конструкторской документации . . . . .	117
84.7 Скрипты и пользовательские расширения . . . . .	117

<b>85 Эксплуатация станочного оборудования</b>	<b>118</b>
<b>86 Основы ЧПУ и цифрового производства</b>	<b>119</b>
86.1 САМ-пакеты для FreeCAD . . . . .	119
 <b>V Основы теории систем автоматического управления</b>	 <b>120</b>
<b>87 Математический аппарат</b>	<b>121</b>
87.1 Передаточная функция . . . . .	121
87.2 Устойчивость САУ . . . . .	121
87.3 Сети Петри . . . . .	121
87.4 Автоматы Маркова . . . . .	121
 <b>88 Релейное управление</b>	 <b>122</b>
 <b>89 Пропорциональные САУ</b>	 <b>123</b>
 <b>90 ПИДн-регуляторы</b>	 <b>124</b>
 <b>VI Разработка ПО для встраиваемых систем</b>	 <b>125</b>
<b>91 Вспомогательные скрипты на языке Python</b>	<b>126</b>
91.1 Установка под Windows . . . . .	128
91.2 Запуск . . . . .	129

91.3	Дополнительные материалы . . . . .	134
92	Make: управление сборкой проектов	135
93	VCS: системы контроля версий	136
93.1	CVS . . . . .	136
93.2	Subversion . . . . .	136
93.3	Git . . . . .	136
93.3.1	GitHub . . . . .	136
94	Основы Си и $C_+^+$	137
94.0.2	Установка MinGW (win32) . . . . .	137
94.1	Особенности $C_+^+$ в embedded . . . . .	137
95	LLVM и разработка собственных компиляторов	138
95.1	Лексический и синтаксический анализ . . . . .	138
95.2	Применение flex/bison для разбора текстовых форматов данных . . . . .	138
95.3	Компилятор Паскаля . . . . .	138
96	Сборка кросс-компилятора GNU toolchain	139
VII	Микроконтроллеры Cortex-Mx	140
97	Отладочные платы	141
97.1	STM32DISCOVERY /Cortex-M3 STM32F103/ . . . . .	141

97.2 STM32F4DISCOVERY /Cortex-M4 STM32F407/ . . . . .	141
<b>VIII Периферия</b>	<b>142</b>
<b>IX Встраиваемый emLinux</b>	<b>143</b>
98 cross	144
99 BuildRoot	145
100 Особенности OpenWrt	146
101 Библиотека SDL	147
101.1 Реализация microGUI . . . . .	147
102 Приложения для X Window	148
103 Программирование сетевых приложений	149
104 Сборка кросс-компилятора GNU мальтийским крестом	150
<b>X IDE</b>	<b>151</b>
105  eclipse	154

105.1	Проверка орфографии	156
<b>106</b>	<b>Code::Blocks</b>	<b>159</b>
<b>107</b>	<b>(g)Vim</b>	<b>160</b>
107.1	Установка под $\boxplus$ Windows	162
107.2	Выход из (g)Vim	165
107.2.1	Выход с автосохранением	165
107.3	Переход в режим редактирования	165
107.4	Переход в режим команд	165
107.5	Запись редактируемого файла	166
107.6	Перезагрузка файла	166
107.7	Отмена последних изменений (undo)	166
<b>XI</b>	<b>Замечания для авторов</b>	<b>167</b>
107.8	Набор репозиторий на GitHub	168
107.9	Верстка в $\LaTeX$	168
<b>XII</b>	<b>Подготовка публикаций в <math>\LaTeX</math></b>	<b>170</b>
107.10	Установка MikTeX под $\boxplus$ Windows	173
107.11	Структура документа	173
107.11.1	Ваголовочный файл или блок	173
107.11.2	Стили документа	173
107.11.3	Пакеты	173

107.11.4	Автор и название	173
107.11.5	Верстка титульных страниц	173
107.11.6	Оглавление	173
107.1	Верстка слайдов	173
107.1	Список литературы и цитирование	173
107.1	Команды секционирования: часть, глава, раздел,...	176
107.1	Таблицы	176
107.1	Формулы	176
107.1	Перекрестные ссылки и гиперссылки	176
107.1	Истинги скриптов и текстовых данных	176
107.1	Подготовка иллюстраций	176
107.19.	Графики GNUPLOT	177
107.19.	Схемы и графы в GraphViz	177
107.19.	PGF/TikZ	178
107.19.	GLE	178
107.2	Верстка электронных изданий	179

### XIII Куча 180

#### Список литературы 181

## Введение

Первоначально этот материал задумывался как комплект документации к платам BlackSwift и VoCore, но постепенно превратился в толстенный учебник для студентов ВУЗов и научных работников по специализациям, связанным с применением цифровой электроники и компьютерной техники.

Большой упор был сделан на использование открытого некоммерческого программного обеспечения, с целью удешевления учебного процесса, уменьшения себестоимости ваших проектов<sup>1</sup>, и стимулирования вашего участия в развитии этих программных пакетов.

Лицензия на эту книгу пока не выбрана, так что она пока просто пишется в духе **OpenSource**: любой может использовать ее часть, изменять или дополнять, до тех пор, пока не накладываются какие-либо административные, финансовые или юридические ограничения на распространение и развитие оригинальной версии или ее открытых форков.

Приглашаем всех желающих участвовать в развитии этого учебного пособия на форум [ruOpenWrt](#), нам нужна обратная связь по качеству материала, результаты тестирования на вас или ваших студентах, дополнения и замечания.

Мы признательны Bill Collis за разрешение использовать материалы его книги «[An Introduction to Practical Electronics, Microcontrollers and Software Design](#)» [1] в русскоязычном варианте «Азбуки» (Часть I), и конечно он вполне заслуженно включен в основные соавторы этой книги.

---

<sup>1</sup> вряд ли ли у вас окажется лишняя пачка килобаксов на покупку пары коммерческих САПР, по крайней мере пока ваш стартап не взлетит в Top\$100K



# Часть I

## Введение в практическую электронику

Эта часть основана на книге:

**An Introduction to Practical Electronics, Microcontrollers and Software Design**

Second Edition, 01 May-2014

© Bill Collis

[www.techideas.co.nz](http://www.techideas.co.nz)

Мы признательны автору за разрешение использовать материалы его книги в русскоязычном варианте «Азбуки», и конечно он вполне заслуженно включен в основные соавторы этой книги.

We are grateful to the author for permission to use materials of his book in the russian version of «Azбуka», and of course he was deservedly included in the main co-authors of this book.

From: Bill Collis <Bill.Collis@.....nz>  
Date: 2014-11-24 0:53 GMT+04:00  
Subject: Electronis Book  
To: "dponyatov@gmail.com" <dponyatov@gmail.com>

Hi Dmitry

thanks for your email.

I am looking at the future of the book myself and thinking I will open source it. If you will only be in using it in Russian language then that is ok and you need to reference the original book.

Thanks

Bill

# Глава 1

## 1 Введение в практическую электронику 13

Эта книга ©<sup>1</sup> имеет слеующий ряд основных направлений:

- Распознавание электронных компонентов и их правильное использование
- Нарботка цельного набора компетенций по основам электроники
- Использование макетных плат
- Навыки ручной пайки

---

<sup>1</sup> оригинал: [\[1\]](#) B.Collis The Introduction to Practical Electronics. . .

- Использование закона Ома для выбора токоограничивающих резисторов
- Делитель напряжения
- Использование EDA CAD<sup>2</sup> для разработки и подготовки производства печатных плат
- Программирование микроконтроллеров и их сопряжение с внешними устройствами
- Использование транзистора в режиме ключа
- Теория источников питания
- Принципы и схемы электропривода
- Навыки отладки схем, их тестирования и испытаний
- Следование принципам обучения через практику
- Безопасные приемы работы

## 1.1 Ваше обучение по специальности «Технология»

### 1.1.1 Цели обучения технологиям Ново-Зеландской программы

- Технологическая практика

---

<sup>2</sup> [E]lectronic [D]esign [A]utomation, САПР автоматизации проектирования электроники

- **Четкость:** разработка ясных описаний для ваших технологических проектов.
- **Планирование:** думать прежде чем делать, и использовать во время работы документацию: блок-схемы, принципиальные схемы, чертежи разводки плат, диаграммы и эскизы.
- **Наработка навыков:** сборка, отладка и тестирование электронных схем, проектирование и изготовление печатных плат, написание программ для микроконтроллеров.

- **Технологические знания**

- **Моделирование:** прежде чем строить готовое электронное устройство, сначала важно понять как оно работает путем моделирования и/или макетирования аппаратного и программного обеспечения.
- **Технологические продукты:** знания о компонентах и их характеристиках.
- **Технологические системы:** электронное устройство является более, чем набором компонентов, это функционирующая система с входами, выходами и контролирующим процессом.

- **Природа технологии**

- **Значение технологических достижений:** знания об электронных компонентах, особенно микроконтроллерах, как основе современных технологий.
- **Роль технологии в обществе:** электронные устройства в настоящее время играют центральную роль в инфраструктуре нашего современного общества; подчинили ли они нас себе, как они изменили нашу жизнь?

## 1.2 Ключевые компетенции Ново-Зеландской программы

- **Знания:** для меня предметом технологии является все что относится к знанию. Моя цель: заставить студентов понимать технологии, заложенные в электронные устройства. Для достижения этого понимания студенты должны активно учиться<sup>3</sup> в работе на самом раннем этапе, чтобы они могли построить собственное понимание предмета и пойти дальше, чтобы стать хорошими решателями проблем. В начале обучения электронике это требует от студентов восприимчивости к инструкциям, которые им дают, и поиск ясности, когда они не понимают их.

Для этого на занятиях рассматриваются много новых и различных элементов знаний, и студентам выдаются задания на решение проблем, чтобы помочь им мыслить логически. Копирование чужого ответа наказывается, но приветствуется совместная работа. В основе обучения лежит построение правильных концептуальных моделей и анализ в контексте "большой картины".

- **Взаимодействие:** работа в парах и группах, это важно как в классе, так и в любой другой ситуации в жизни; мы все должны договариваться и разделять ресурсы и оборудование с другими людьми; поэтому крайне важно активное общение и помощь друг другу.
- **Использование языка символов и текстов:** сердцем нашего предмета является язык, который мы используем для обмена информацией в электронных схемах, планах, алгоритмах и синтаксисе компьютерных языков программирования; так что способность распознавать и правильно использовать символы и диаграммы для работы, которую мы делаем, имеет критическое значение.
- **Самоконтроль:** студенты принимают на себя личную ответственность за собственное обучение; они принимают вызов, надеясь найти ответы в книгах или найти учителя, способного объяснить им, что делать. Это значит, что студенты должны взаимодействовать с рабочим материалом.

---

<sup>3</sup> в оригинале **enage**, англо-калька с *себуанского*, NZ

Иногда ответы приходят легко, иногда нет; часто наша тема требует много проб и ошибок (в основном ошибок). Студенты должны знать, что у них будут трудные времена, пока не будет изучена большая часть. И не сдаваться в поиске понимания.

- **Участие и содействие:** мы живем в мире, который невероятно зависит от технологии, особенно электроники; студенты должны развивать осознание важности этой области человеческого творчества в нашей повседневной жизни, и понимать, что наши проекты имеют и социальную функцию, а не только техническую.

## Глава 2

### 2 Вводная электронная схема 15



## Глава 3

### 3 Вводное конструирование печатной платы 26

## Глава 4

### 4 Пайка, припой и паяльники 41

## Глава 5

### 5 Введение в теорию электроники 49

## Глава 6

### 6 Введение в электронику микроконтроллера 63

## Глава 7

### 7 Входные цепи микроконтроллера 91

## Глава 8

### 8 Обзор программирования 104

## Глава 9

### 9 Введение в поток выполнения программы 112

## Глава 10

10 Вводное программирование:  
использование подпрограмм 126



# Глава 11

11 Вводное программирование:  
Использование переменных 134

## Глава 12

### 12 Основные дисплеи 161

## Глава 13

### 13 Проект портативного аудиоусилителя на TDA2822M 174

## Глава 14

### 14 Основы логического программирования 187

## Глава 15

### 15 Разработка алгоритма: Система сигнализации 202

## Глава 16

### 16 Основы теории цепей постоянного тока 215

## Глава 17

### 17 Основы планирования проекта 236

## Глава 18

18 Пример дизайна системы: Таймер  
клеевого пистолета 268



## Глава 19

### 19 Основные интерфейсы и их программирование 273

## Глава 20

### 20 Основы интерфейса аналого-цифрового преобразования 295

## Глава 21

### 21 Основы проектирования системы 314

## Глава 22

22 Основы проектирования системы:  
Тайм-трекер 317

## Глава 23

### 23 Основы вычислений времени 330

## Глава 24

### 24 Основы строковых переменных 340

## Глава 25

### 25 Силовые интерфейсы 353

## Глава 26

### 26 Теория источников питания 370



## Глава 27

27 Типичные вопросы тестирования  
2011/12/13 годов 395

## Глава 28

### 28 Расширенное программирование: Массивы 397

## Глава 29

### 29 Подтягивающие резисторы AVR 402

## Глава 30

### 30 Дополнительноподключение клавиатуры 403

## Глава 31

### 31 Тонкости циклов Do-Loop & While-Wend 417

## Глава 32

### 32 Подключение двигателя постоянного тока 423

## Глава 33

### 33 Пример расширенной системы: Будильник 452

## Глава 34

### 34 Резистивный сенсорный экран 468



## Глава 35

### 35 Пример проектирования системы: Регулятор температуры 475

## Глава 36

36 Расширенное программирование:  
Машины состояний 478

## Глава 37

### 37 Переработанный проект будильника 501

## Глава 38

38 Студенческий проект: Расширенный  
оконный контроллер 514

## Глава 39

### 39 Альтернативные техники кодирования машин состояния 524

## Глава 40

40 Сложно: последовательная связь 526

## Глава 41

### 41 Цифровой радиоканал 597

## Глава 42

### 42 Введение в I2C 617



## Глава 43

43 Студенческий проект: Таймер полива теплицы 631

## Глава 44

### 44 Проект Велосипедного аудиоусилителя 642

## Глава 45

### 45 Графические LCD 648

## Глава 46

### 46 Проект Отслеживания температуры GLCD 660

## Глава 47

### 47 Прерывания 672

## Глава 48

48 Таймеры/Счётчики 692

## Глава 49

49 Проект скроллинга графического LED  
дисплея: массивы и таймеры 698

## Глава 50

50 Проект медицинского прибора:  
реализация таймера 709



## Глава 51

Проект часов на 7-сегментном  
индикаторе  
реализация на сдвоенном таймере

## Глава 52

52 ИС драйвера дисплея MAX 7219/7221  
739

## Глава 53

53 Подключение через мобильную связь:  
ADH8066 744

## Глава 54

### 54 Передача данных через Internet 778

## Глава 55

### 55 Задание: математика в реальном мире 816

## Глава 56

### 56 Цветной графический LCD на основе SSD1928 825

## Глава 57

57 Светофор: помощь и решение 865

## Глава 58

58 Компьютерное программирование:  
низкоуровневые детали 869



## Глава 59

### 59 USB-программатор: USBASP 876

## Глава 60

### 60 Программатор USBTinyISP 877

# Глава 61

## 61 Программирование на Си и AVR 881

## Глава 62

### 62 Объектно-Ориентированное Программирование (ООП) на $C_{+}^{+}$ и AVR 929

## Глава 63

### 63 Современные (2014) отладочные платы на AVR 953

## Глава 64

64 Eagle: создание собственной  
библиотеки 970

## Глава 65

### 65 Практические методы 979

Глава 66

66 ЧПУ 990



Глава 67

67 Индекс 1008

# Часть II

## Основы электроники

Здесь идет список ссылок на онлайн лекции в edX, Coursera, и т.п.

## Глава 68

# Линейные схемы на пассивных элементах, основы электротехники

## Глава 69

# Симуляция и расчет схем в ngSPICE

# Глава 70

## KiCAD

70.1 Отрисовка схем в KiCAD

70.2 Библиотеки элементов

70.3 Передача схемы в ngSPICE

# Глава 71

## Простейшие полупроводниковые элементы

71.1 Оптоэлектроника

71.2 Схемы на биполярных транзисорах

71.3 Схемы на на полевых транзисорах

## Глава 72

# Операционные усилители



# Глава 73

## Источники питания

73.1 Батарейное питание

73.2 Линейные стабилизаторы

73.3 Импульсные преобразователи на ШИМ-контроллерах

73.4 Цепи защиты и гашения кондуктивных помех

# Глава 74

## Цифровая электроника



## Глава 75

# Компьютерные интерфейсы

### 75.1 Поколение 90х: COM, LPT, ISA

#### 75.1.1 Резервный программатор AVR “пять проводков”

### 75.2 Сеть CAN

### 75.3 Интерфейсные модули USB

#### 75.3.1 Универсальный высокоскоростной конвертер FTDI FT2232H

#### 75.3.2 JTAG-адаптер

#### 75.3.3 Отладочный модуль CAN

Глава 76

ПЛИС

# Глава 77

## Датчики

## Глава 78

# Электропривод и исполнительные устройства

## Часть III

# Основы конструирования РЭС



## Глава 79

# Пакеты моделирования на основе OpenFOAM

## Глава 80

# Обеспечение теплового режима

# Глава 81

## Электромагнитная совместимость

81.1 Кондуктивные помехи

81.2 Компоновочные модели и оптимизация кабельной сети

# Часть IV

## Технология РЭС

# Глава 82

## Инструменты и оборудование

### 82.1 JTAG-адаптер

### 82.2 Отладочные платы

Прежде чем начать работать с отдельными МК, устанавливая их на плату собственной разработки, для быстрого старта используют *отладочные платы*<sup>1</sup>

---

<sup>1</sup> development board, demo board

### 82.2.1 Arduino /Atmel Mega AVR8/

### 82.2.2 Cortex-Mx

См. [97](#)

82.2.3 CubieBoard /Cortex-A8 AllWinner A10/

82.2.4 Raspberry Pi /ARM11 BCM3032/

82.2.5 BlackSwift /MIPS/

82.2.6 VoCore /MIPS/

82.3 Монтажный инструмент

82.4 Измерительное оборудование

82.4.1 Тестер

82.4.2 Осциллограф

82.4.3 Логический анализатор

82.4.4 Генератор сигналов

82.4.5 Рыльцесметр

## 82.5 Электроинструмент

### 82.5.1 Дрель



Дрель ударная сетевая  
Praktyl-R PID13D01 400 Вт  
(!)395 р.



Дрель безударная сетевая  
Интерскол Д-11/5303Р (с БЗП)  
1120 р.



## Глава 83

# Трассировка плат и подготовка производства в KiCAD

83.1 Технология ЛУТ (Лазерный УТюг)

83.2 Технология фоторезиста

83.3 Формат Gerber и подготовка промышленного производства



# Глава 84

## FreeCAD



1

В среде специалистов ряда отраслей известна проблема создания полноценной САПР в рамках OpenSource, и хотя FreeCAD ещё не является кандидатом на такую «полноценность», этот продукт может рассматриваться как одна из попыток создания базы для решения этой проблемы. Разработчик FreeCAD Юрген Ригель, работающий в корпорации DaimlerChrysler, позиционирует свою программу как первый бесплатный инструмент проектирования механики (сравнивая свой продукт с такими развитыми проприетарными системами как CATIA версий 4 и 5, SolidWorks), созданный на основе библиотеки **Open CASCADE**. Цель программы — предоставить базовый инструментарий этой библиотеки в интерактивном режиме.

Следует отметить, что имеет место ещё один программный продукт имеющий название freeCAD, его разработчик — Aik-Siong Koh, и он не связан с FreeCAD'ом Юргена Ригеля.

## 84.1 Установка под ☐Windows



<sup>1</sup> копипаста [https://ru.wikipedia.org/wiki/FreeCAD\\_\(Juergen\\_Riegel%27s\)](https://ru.wikipedia.org/wiki/FreeCAD_(Juergen_Riegel%27s))



84.2 Чертеж

84.3 Эскиз

84.4 Деталь

84.5 Сборка

84.6 Автогенерация конструкторской документации

84.7 Скрипты и пользовательские расширения

## Глава 85

# Эксплуатация станочного оборудования

## Глава 86

# Основы ЧПУ и цифрового производства

### 86.1 CAM-пакеты для FreeCAD



## Часть V

# Основы теории систем автоматического управления

# Глава 87

## Математический аппарат

87.1 Передаточная функция

87.2 Устойчивость САУ

87.3 Сети Петри

87.4 Автоматы Маркова

## Глава 88

# Релейное управление

## Глава 89

### Пропорциональные САУ

# Глава 90

## ПИДn-регуляторы

## Часть VI

# Разработка ПО для встраиваемых систем

# Глава 91

## Вспомогательные скрипты на языке Python



*Название языка произошло вовсе не от вида пресмыкающихся. Автор назвал язык в честь популярного британского комедийного телешоу 1970-х «Летающий цирк Монти Пайтона». Впрочем, всё равно название языка чаще ассоциируют именно со змеей, нежели с передачей — пиктограммы файлов в KDE или в Microsoft Windows и даже эмблема на сайте <http://www.python.org> (до выхода версии 2.5) изображают змеинные головы.*

Python<sup>1</sup> — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода.

Python удобно применять для написания различных вспомогательных скриптов. Часто его используют при разработке сложных программных систем для написания первых версий. В процессе работы над большими программами часто перерабатываются большие объемы кода, поэтому для ускорения разработки требуется максимально высокоуровневый язык. После того как архитектура программы стабилизируется, узким местом становится производительность, и программу переписывают на более низкоуровневом компилируемом языке, чаще всего  $C^+$ .

Написание программ упрощают:

- **объектно-ориентированное программирование** облегчает разработку программ, позволяет переопределить стандартные операторы для пользовательских типов данных, упрощая синтаксис
- **динамическая типизация** не требуется заранее упределять переменные, они создаются простым присваиванием
- **обработка исключений** для секции кода можно определить обработчик ошибок
- **высокоуровневые структуры данных** — списки, словари (набор элементов ключ:значение), очереди
- богатая стандартная библиотека и множество дополнительных библиотек на все случаи

---

<sup>1</sup> в оригинале читается **пáйтон**, но давно русифицировался как **питóн**

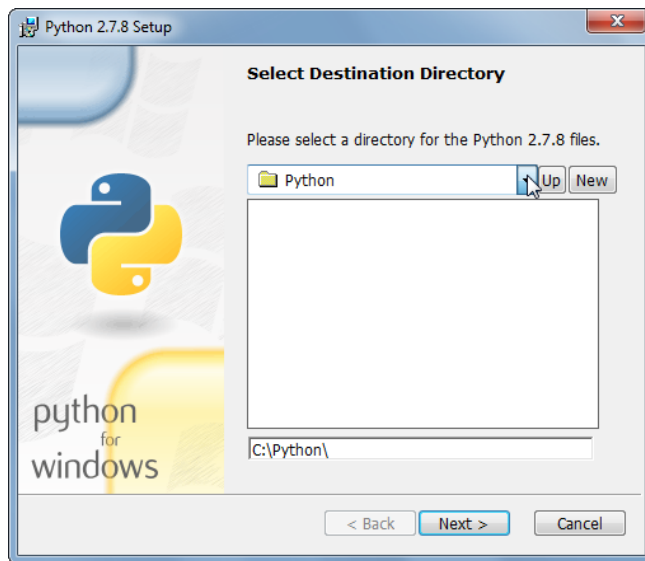
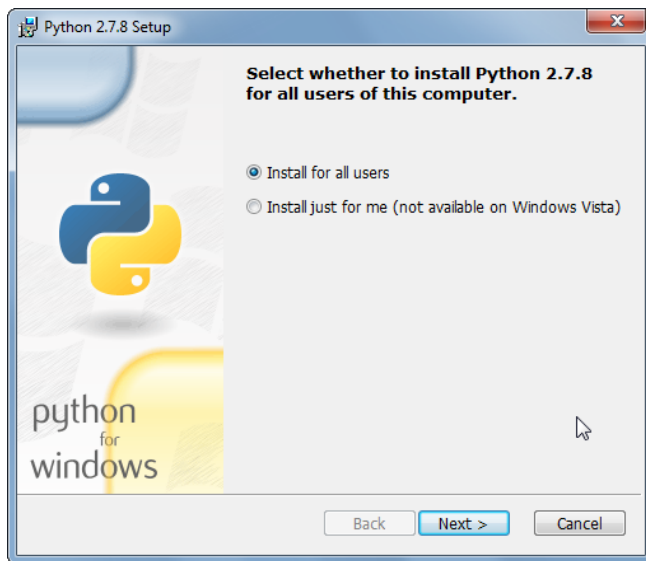


## 91.1 Установка под Windows

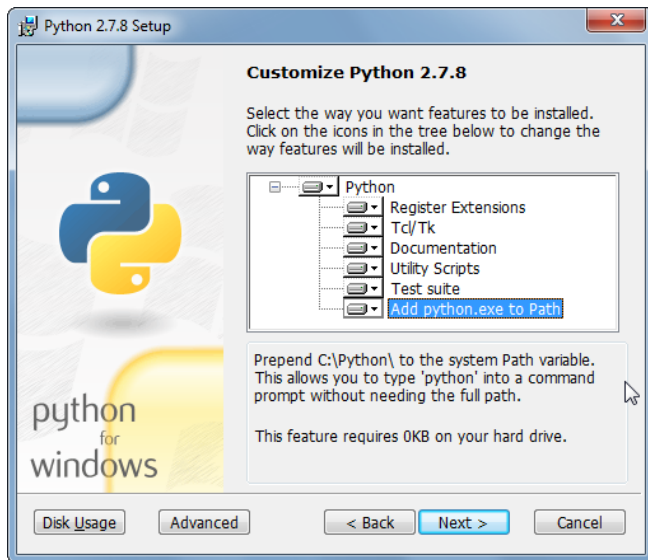
⊞ + R >> <http://www.python.org> >> Downloads >> Python 2.7.8

python-2.7.8.msi >> Setup >> for all users/for me

Destination Directory >> C:/Python/ >> Next

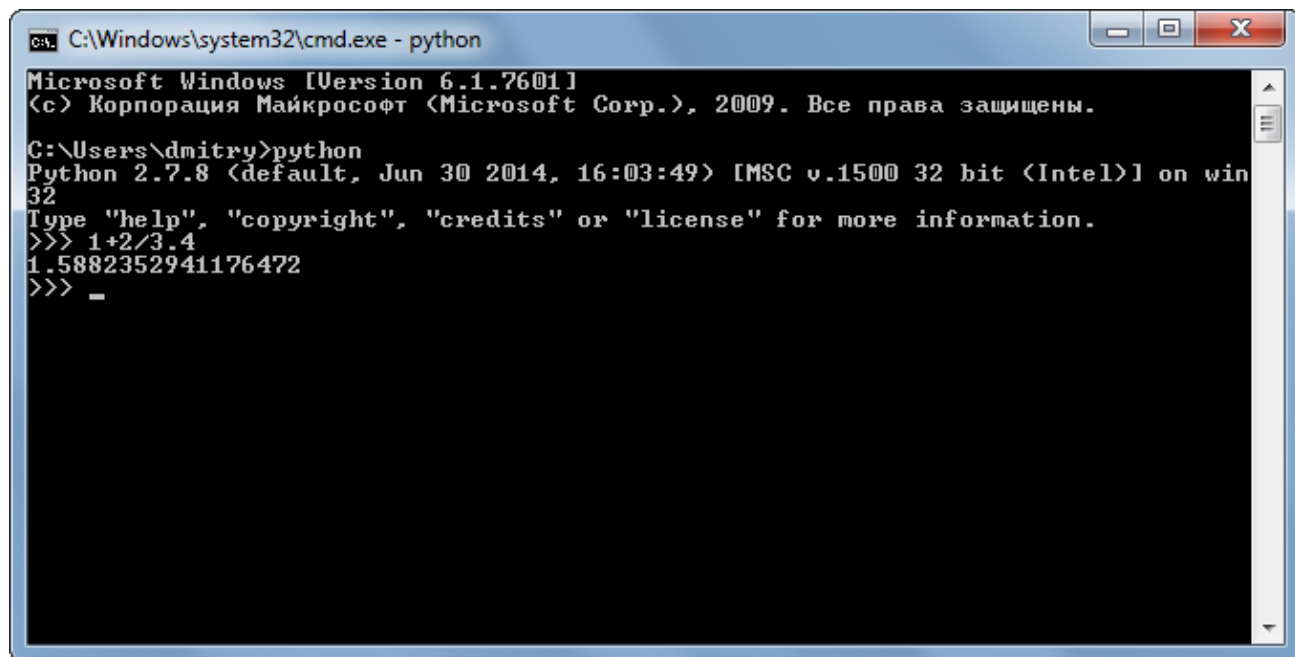


Customize >> Python >> Add python.exe to PATH >> Next >> Finish



## 91.2 Запуск

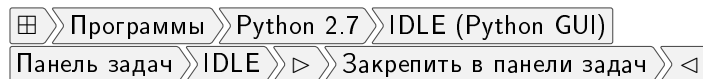
Из командной строки:  +  cmd >> python



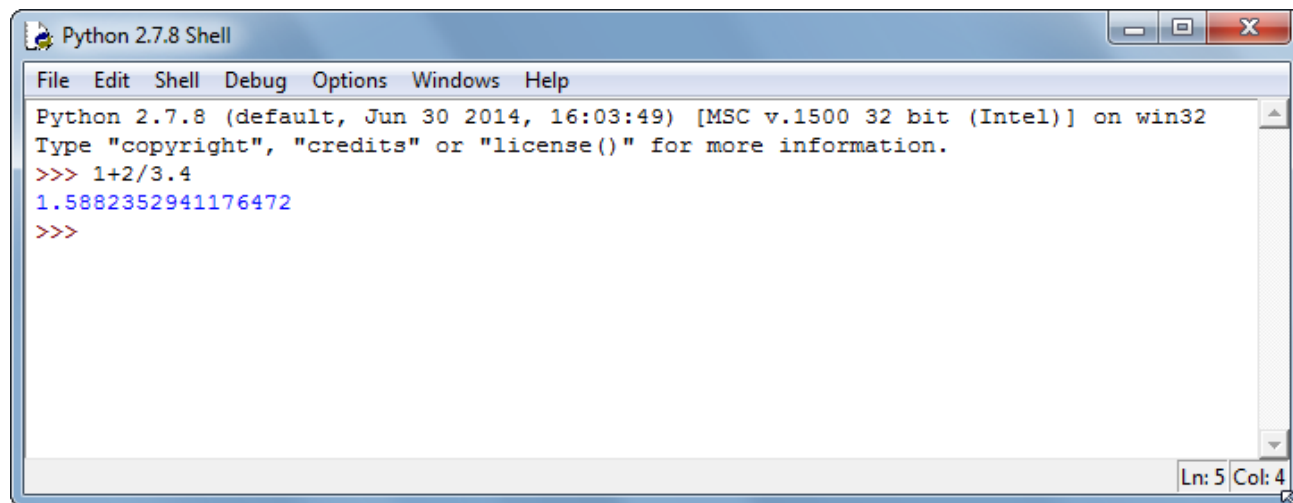
```
C:\Windows\system32\cmd.exe - python
Microsoft Windows [Version 6.1.7601]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.

C:\Users\dmitry>python
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "help", "copyright", "credits" or "license" for more information.
>>> 1+2/3.4
1.5882352941176472
>>> _
```

Простейшая среда IDLE<sup>2</sup>:



<sup>2</sup> на GUI-библиотеке Tkinter, идущей в комплекте



A screenshot of a Windows command prompt window titled "Python 2.7.8 Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The text inside the window shows the Python version and build information: "Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win32". It then prompts the user to type "copyright", "credits", or "license()". The user has entered ">>> 1+2/3.4", and the shell has responded with the result "1.5882352941176472". The prompt ">>>" is shown again on the next line. The status bar at the bottom right indicates "Ln: 5 Col: 4".

```
Python 2.7.8 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 1+2/3.4
1.5882352941176472
>>>
```

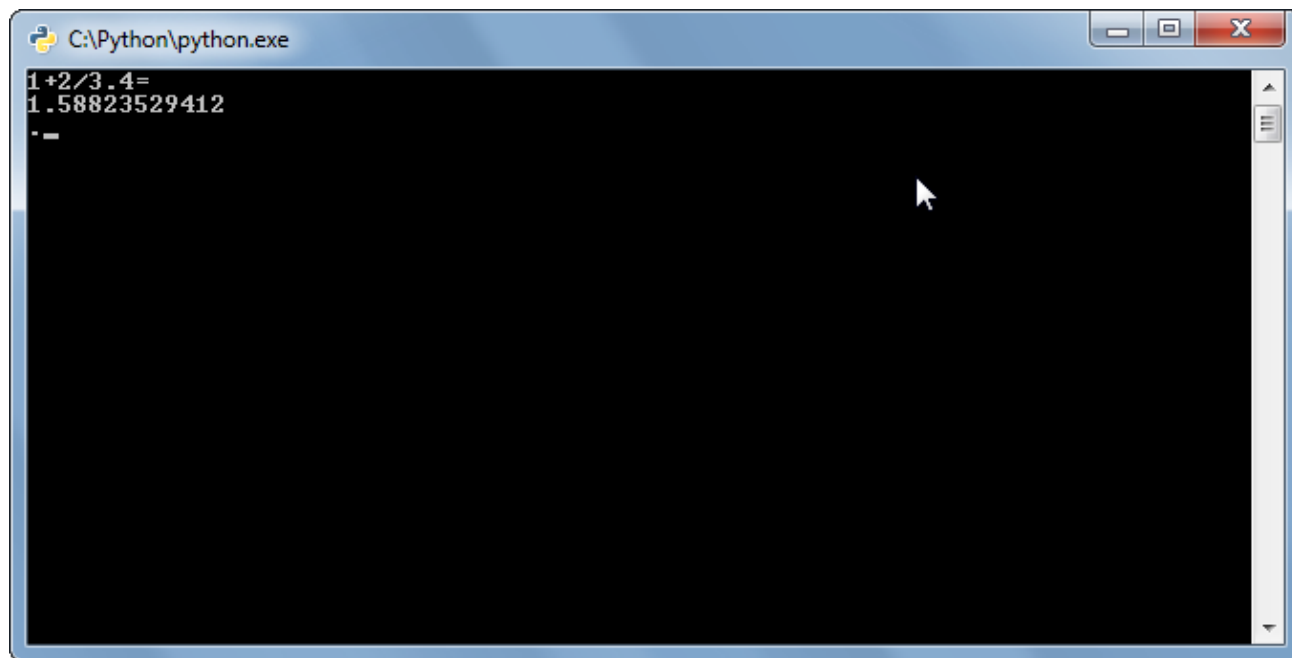
◀◀ по файлу скрипта:

 +  notepad /tmp/py.py

/tmp/py.py

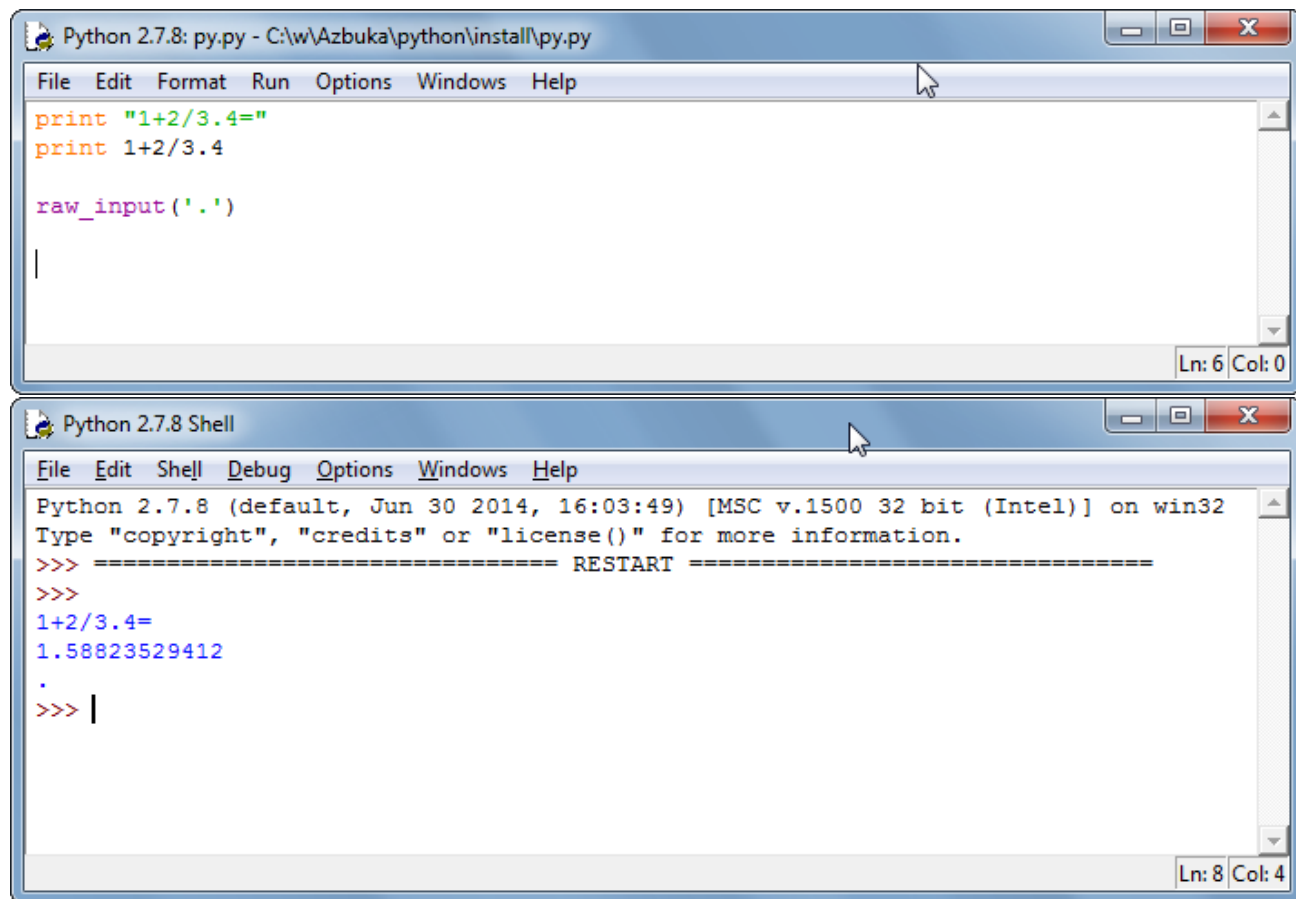
```
1 print "1+2/3.4="
2 print 1+2/3.4
3
4 raw_input( '. ' )
```

 +  /tmp/py.py



Открытием файла скрипта в IDLE:





## 91.3 Дополнительные материалы

- [4] Г. Россум, Ф.Л.Дж. Дрейк, Д.С. Откидач, [Язык программирования Python](#)
- [3] Аллен Дауни [Думать на языке Python: Думать как компьютерный специалист](#)

## Глава 92

Make: управление сборкой проектов



## Глава 93

# VCS: системы контроля версий

### 93.1 CVS

### 93.2 Subversion

### 93.3 Git

#### 93.3.1 GitHub

# Глава 94

## Основы Си и $C_{+}^{+}$

94.0.2 Установка MinGW (win32)

94.1 Особенности  $C_{+}^{+}$  в embedded

## Глава 95

# LLVM и разработка собственных компиляторов

95.1 Лексический и синтаксический анализ

95.2 Применение flex/bison для разбора текстовых форматов данных

95.3 Компилятор Паскаля

## Глава 96

# Сборка кросс-компилятора GNU toolchain

## Часть VII

# Микроконтроллеры Cortex-Mx

# Глава 97

## Отладочные платы

97.1 STM32DISCOVERY /Cortex-M3 STM32F103/

97.2 STM32F4DISCOVERY /Cortex-M4 STM32F407/

# Часть VIII

## Периферия

# Часть IX

## Встраиваемый emLinux



# Глава 98

cross

Глава 99

BuildRoot

# Глава 100

## Особенности OpenWrt

# Глава 101

## Библиотека SDL

### 101.1 Реализация microGUI

# Глава 102

## Приложения для X Window

# Глава 103

## Программирование сетевых приложений

## Глава 104

# Сборка кросс-компилятора GNU мальтийским крестом

Часть X

IDE



IDE — Integrated Development Environment, интегрированная среда разработки.  
Программный пакет, включающий



- средства управления проектом,
- отслеживание зависимостей между файлами (в т.ч. с анализом исходного текста программ на конструкции типа `#include`, `module`, `uses`),
- автозапуском компиляторов для изменившихся файлов,
- GUI для отладчиков (`gdb`),
- специализированный редактор plain text<sup>1</sup> файлов с
  - цветовой и шрифтовой *подсветкой синтаксиса*,
  - *автодополнением*: дописываются имена объектов программ, синтаксические конструкции и параметры функций,
  - *автоформатированием*: фрагмент текста переформатируется в соответствии с синтаксисом языка редактируемого файла, проставляются отступы в зависимости от вложенности синтаксических конструкций типа циклов и условных блоков)
  - выделением строк, на которые указывают сообщения об ошибках компиляторов,
  - маркеры точек останова отладчика
- отображение структуры программ, например деревья классов и структур данных

---

<sup>1</sup> файлы не включающие непечатаемых символов и бинарных данных, которые можно причитать простым выводом на экран командами типа **type**, **cat**, **more**

- контекстные справочники по используемым языкам программирования, автоматический вывод списка параметров при вводе имени функции
- отображение дизассемблерных листингов для компилируемых языков
- отображение браузера как вкладки или MDI окна
- отображение вывода *статических анализаторов* программ с кликабельными ссылками
- вывод компиляторов и трансляторов с цветовым выделением и переход на ошибочную строку в редакторе при щелчке на ошибке
- ...

В этой книге рассмотрены три бесплатных мультиплатформенных OpenSource IDE, в порядке навороченности, универсальности, и требуемым ресурсам для работы самой среды:

1.  ECLIPSE [105](#): самая навороченная и ресурсоемкая IDE, написана на Java, имеет десятки дополнительных модулей на все случаи, умеет работать со всеми распространенными языками программирования, жрет память, и требует современного компьютера минимум с 2+ Гб ОЗУ. Последний релиз  ECLIPSE Luna работает заметно быстрее (особенно при запуске).
2. Code::Blocks [106](#): легкая среда для разработки на C/C<sub>+</sub>, для других языков может потребоваться написать свои модули или файлы описания синтаксиса
3. (g)Vim [107](#): самый легкий и *портатбельный* универсальный текстовый редактор с расширенными функциями, работает на всех существующих платформах (кроме совсем уж embedded), использует минимум ресурсов, но требует некоторого обучения даже чтобы выйти из vim ☺



## Глава 105

☉eclipse



## 105.1 Проверка орфографии

1

То, что проверка орфографии очень удобная вещь вряд ли нужно объяснять. Есть конечно люди, которые не обращают на неё внимание, но это чаще всего из-за экономии времени и отсутствия удобных средств проверки.

Действительно, удобная автоматическая проверка орфографии есть в офисных пакетах, но мне сложно представить разработчика, который будет переносить комментарии в Word и обратно ☺.

Поэтому очень удобно иметь *проверку правописания прямо в IDE*. И ☹ECLIPSE в этом смысле полностью соответствует ожиданиям.

Долго объяснять что к чему нет смысла. Проверка орфографии встроена в ☹ECLIPSE и если вы пишете только на английском, то может быть не захотите ничего менять.

Кроме того, есть **статья Aaron'a** (en) в которой автор рассказывает о подключении дополнительных словарей и плагине **eSpell**.

Но *русских словарей в дистрибутиве нет*, а при подключении внешних есть нюансы. Поэтому мы максимально подробно рассмотрим *подготовку и добавление русских словарей*.

Первый вопрос. В каком виде должны быть словари и где их взять?

Тут всё просто. Формат словаря — обычный текстовый файл, в котором каждое слово начинается с новой строки. И нам вполне подойдут свободно распространяемые словари **aSpell**.

Установка состоит из 4 шагов:

1. качаем aSpell и словари для нужных языков



---

<sup>1</sup> копияста <http://www.simplecoding.org/proverka-orfografii-v-eclipse.html>

Binaries >> Full installer

Precompiled dictionaries >> English

Precompiled dictionaries >> Russian

2. устанавливаем сначала **aSpell**, потом отдельно каждый словарь

**Aspell-0-50-3-3-Setup.exe** >> Setup GNU Aspell >> Next >> License >> Next

Directory >> **C:/GnuWin32/Aspell** >> Next >> Next

Additional >> Next >> Install >> Next >> ☐ View manual >> Finish

**Aspell-en-0.50-2-3.exe** >> Aspell English Dictionary >> Next >> License >> Next

Directory >> **C:/GnuWin32/Aspell** >> Next >> Next >> Install >> Finish

**Aspell-ru-0.50-2-3.exe** >> Aspell Russian Dictionary >> Next >> License >> Next

Directory >> **C:/GnuWin32/Aspell** >> Next >> Next >> Install >> Finish

3. делаем дампы словарей, перекодировываем из koi8r в utf8 и объединяем

 + R cmd

```
1 cd \GnuWin32\Aspell
2 bin\aspell dump master en > en.dict
3 bin\aspell dump master ru > ru.koi8
4 iconv -f koi8-r -t utf-8 < ru.koi8 > ru.dict
5 copy en.dict + ru.dict enru.dict
```

4. настраиваем *spell-checker* ☰ECLIPSE

☰ECLIPSE > Window > Preferences > Editors > Text editors > Spelling

User defined dictionary > **C:/GnuWin32/Aspell/enru.dict**

Encoding > UTF-8

Apply > OK

Глава 106

Code::Blocks



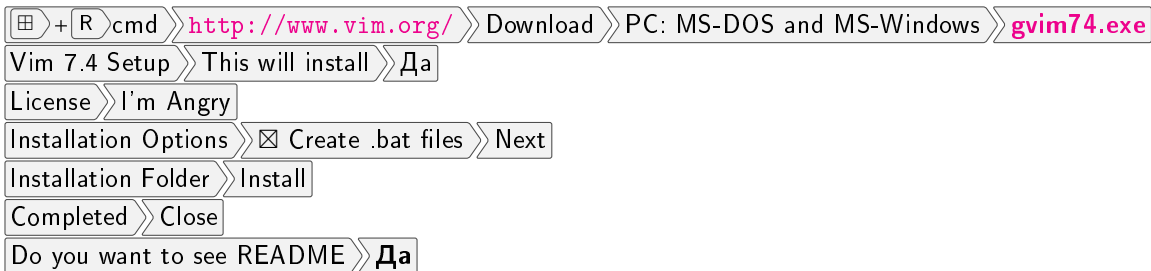


## Глава 107

### (g)Vim



## 107.1 Установка под Windows



Теперь можно настроить темную тему и выключение подсветки синтаксиса, по умолчанию после установки используется светлая тема и подсветка выключена:

меню > Правка > Настройка запуска

Переходим в конец файла и включаем *режим вставки*

Ctrl + Down Ins Enter Enter

```
1 syntax on
2 colorscheme pablo
```

Выходим в *режим команд* и принудительно сохраняем



Esc : w ! Enter Enter

Выходим из (g)Vim

Esc : q ! Enter

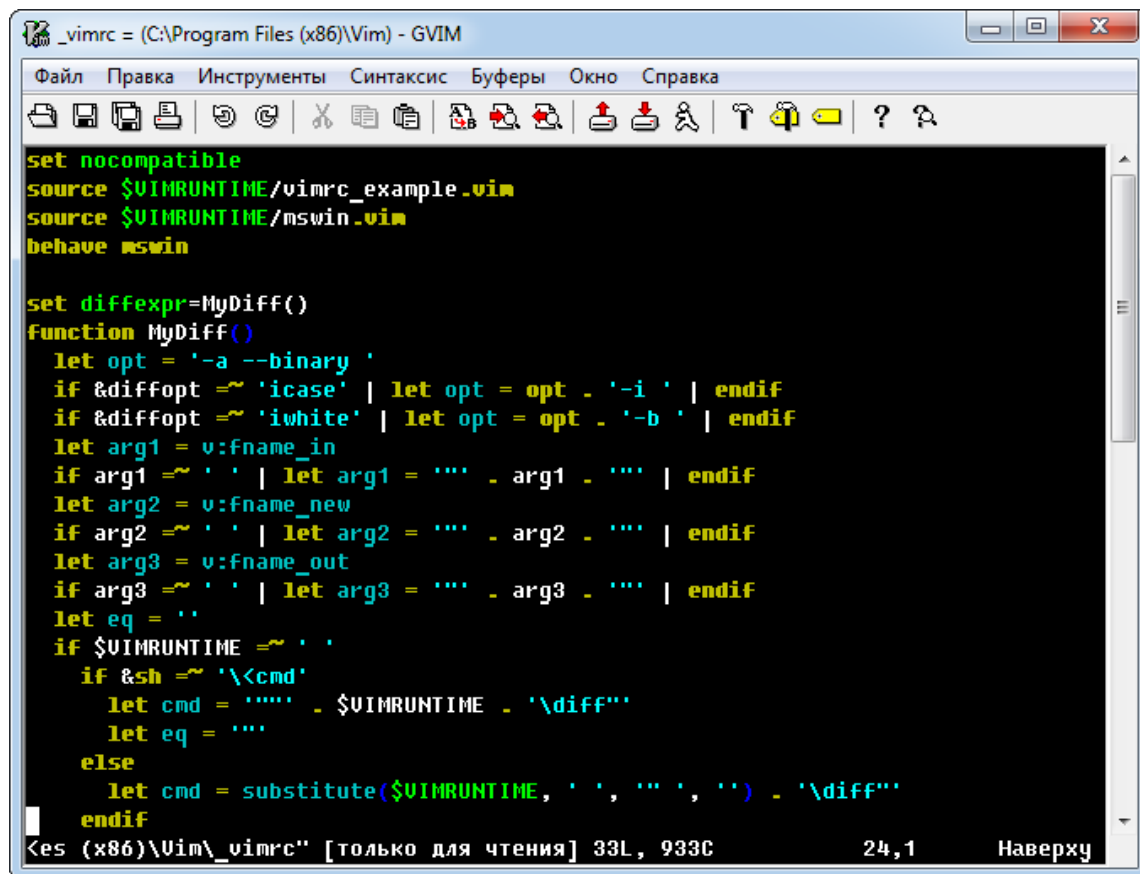
Если не получилось (под Windows 7):

 +  cmd >> /Program Files (x86)/Vim/

Копируем файл **\_vimrc** в любой каталог, например в **/tmp/**, затем   Edit with Vim, и повторяем редактирование еще раз.

Затем копируем **\_vimrc** обратно в **/Program Files (x86)/Vim/** с заменой.

Если теперь открыть на редактирование тот же файл, или любой другой текстовый, получим более удобный вид: для файлов известных типов будет работать подсветка синтаксиса.



The screenshot shows the GVIM editor window titled "\_vimrc = (C:\Program Files (x86)\Vim) - GVIM". The menu bar includes "Файл", "Правка", "Инструменты", "Синтаксис", "Буферы", "Окно", and "Справка". The toolbar contains various icons for file operations, editing, and navigation. The main text area displays the contents of the vimrc file with syntax highlighting: keywords in green, variables in yellow, and strings in red. The code includes settings for 'nocompatible', sourcing runtime files, and a custom 'MyDiff()' function. The status bar at the bottom shows the file path, line/character count, and a "Наверх" button.

```
set nocompatible
source $VIMRUNTIME/vimrc_example.vim
source $VIMRUNTIME/mswin.vim
behave mswin

set diffexpr=MyDiff()
function MyDiff()
  let opt = '-a --binary '
  if &diffopt =~ 'icase' | let opt = opt . '-i ' | endif
  if &diffopt =~ 'iwhite' | let opt = opt . '-b ' | endif
  let arg1 = v:fname_in
  if arg1 =~ ' ' | let arg1 = '"' . arg1 . '"' | endif
  let arg2 = v:fname_new
  if arg2 =~ ' ' | let arg2 = '"' . arg2 . '"' | endif
  let arg3 = v:fname_out
  if arg3 =~ ' ' | let arg3 = '"' . arg3 . '"' | endif
  let eq = ''
  if $VIMRUNTIME =~ ' '
    if &sh =~ '\<cmd'
      let cmd = '""' . $VIMRUNTIME . '\diff"'
      let eq = ''
    else
      let cmd = substitute($VIMRUNTIME, ' ', '" ', '') . '\diff"'
    endif
  endif
endfunction
<es (x86)\Vim\_vimrc" [только для чтения] 33L, 933C      24,1      Наверх
```

## 107.2 Выход из (g)Vim

`Esc` : `!` `q` `Enter`

### 107.2.1 Выход с автосохранением

`Esc` `Shift` + `Z` `Shift` + `Z`

## 107.3 Переход в режим редактирования

(g)Vim запускается в *командном режиме*, для перехода в режим редактирования используются следующие клавиатурные команды:

- `Ins` или `i`: включение *режима вставки* по текущему положению курсора
- `Ins` `Ins` или `r`: включение *режима перезаписи* поверх текста после курсора
- `Shift` + `A`: включение режима вставки *в конец текущей строки*

## 107.4 Переход в режим команд

`Esc`

## 107.5 Запись редактируемого файла

`Esc` : `w` `Enter`

Если выводится предупреждение типа “файл защищен от записи” или подобное, может сработать принудительная запись:

`Esc` : `!` `w` `Enter`

## 107.6 Перезагрузка файла

Для перезагрузки возможно измененного извне файла или отмены всех несохраненных изменений

`Esc` : `e` `Enter`

## 107.7 Отмена последних изменений (undo)

`Esc` `u` `u` ...

# Часть XI

## Замечания для авторов



## 107.8 Набор репозиторий на GitHub

<a href="https://github.com/ponyatov/Azbuka">https://github.com/ponyatov/Azbuka</a>	основная репа
<a href="https://github.com/ponyatov/bib">https://github.com/ponyatov/bib</a>	библиографические базы данных
<a href="https://github.com/ponyatov/scratcher">https://github.com/ponyatov/scratcher</a>	журнал, используются некоторые материалы

Для работы с проектом сделайте собственный форк основной репы, библиографическую базу и журнал можете клонировать напрямую. Создайте каталог и склонируйте репы:

```

1 D:
2 cd \
3 mkdir w
4 cd \w\
5 git clone --depth=1 -o gh git@github.com:username/Azbuka.git
6
7 git clone --depth=1 -o gh git@github.com:ponyatov/bib.git
8 git clone --depth=1 -o gh git@github.com:ponyatov/scratcher.git

```

## 107.9 Верстка в $\text{\LaTeX}$

Было много вопросов по выбору языка разметки, и даже предложения некоторые материалы просто навордять. Но все же используется  $\text{\LaTeX}$ , т.к. это самая навороченная система подготовки больших изданий, широко распространенная (в узких кругах), и прежде всего как имеющая богатейший набор пакетов-расширений для всевозможных вывертов.

$\text{\LaTeX}$  не предназначен для верстки полноцвета, журнальной верстки или ручного таскания блоков по листу.  $\text{\LaTeX}$  изначально был заточен на подготовку научно-технической и учебной многостраничной литературы с логической разметкой.

Как профессиональный инструмент,  $\text{\LaTeX}$  требует обучения. Начерно навордять на нем текст, накидав как попало картинок, и наляпав шрифтов<sup>1</sup> не получится. И это хорошо.

Но — материалы на добавление принимаются в любых форматах, группой авторов, способных их доварить до нужного качества. Единственное ограничение: наличие бесплатных средств просмотра на трех основных платформах:  $\boxtimes$ Windows, Linux и MacOS, или в онлайн (Google Docs, M\$ облака, и прочие сетевые болота).

Также приветствуется использование различных более простых языков разметки (SPHINX, Wiki, DocBook, .md, ...). Для первоначального сбора и группировки материала они проще для освоения, чаще всего рендер-движки для этих языков заточены под веб-редактирование в т.ч. групповое, и хорошо подходят для простой по оформлению документации на программные пакеты, или как сборник ссылок на другие ресурсы.

Для включения таких материалов в основную верстку несложно написать  $\text{\TeX}$ -транслятор (если нет сразу готового), который создаст .tex файлы нужного вида с минимумом ручной доработки.

---

<sup>1</sup> про существование стилей многие вордьятники даже не слышали

## Часть XII

# Подготовка публикаций в L<sup>A</sup>T<sub>E</sub>X

LaTeX (по-русски произносится *латéx*) — наиболее популярный набор макрорасширений (или макро-пакет) системы компьютерной вёрстки T<sub>E</sub>X, который облегчает набор сложных документов. В типографском наборе форматируется как L<sup>A</sup>T<sub>E</sub>X.

Главная идея L<sup>A</sup>T<sub>E</sub>X состоит в том, что авторы должны думать о содержании, о том, что они пишут, не беспокоясь о конечном визуальном облике (печатный вариант, текст на экране монитора или что-то другое). Готовя свой документ, автор указывает логическую структуру текста (разбивая его на главы, разделы, таблицы, изображения), а L<sup>A</sup>T<sub>E</sub>X решает вопросы его отображения. Так содержание отделяется от оформления. Оформление при этом или определяется заранее (стандартное), или разрабатывается для конкретного документа.

В практическом смысле использование L<sup>A</sup>T<sub>E</sub>X позволяет (в порядке уменьшения важности):

- с помощью макросов и T<sub>E</sub>X-программирования реализовывать любые стили и самую сложную верстку, существует множество готовых пакетов для верстки графических химических формул, разнообразных схем, транскрипционных знаков, внезапно электронных схем, цветных листингов и т.п.
- автоматизировать работу с документами: пересобирать выходные файлы через [Make](#), генерировать части документов с помощью своих скриптов<sup>3</sup>
- получить выходной документ в .pdf .html .txt .PostScript .djvu ... с кликабельными ссылками, анимированными, а иногда и интерактивными элементами
- не использовать файлы документов в закрытом формате

---

<sup>2</sup> копияста <https://ru.wikipedia.org/wiki/LaTeX>

<sup>3</sup> отчеты, стандартные формы, результаты работы любых программ

- легко держать набор файлов в [VCS](#)
- не покупать текстовый процессор

Особенно важен пункт про сложную верстку: она всегда нужна в крупных технических публикациях, особенно в учебной литературе, или отчетных работах. Вам обязательно понадобится вставлять графики экспериментальных данных, тематически специфичные схемы, листинги, выходные данные работы ваших программ и т.п.

Традиционно  $\text{\LaTeX}$  любим математиками, и всеми кто готовит публикации с большим количеством формул и перекрестных ссылок: после небольшого обучения формулы вводятся с листа со скоростью набора текста, особенно если ваш редактор умеет автодополнение, и никакой мышью возни.

Естественно всякие чисто автоматические вещи типа автонумерации ссылок и формул, сборки оглавлений и индексов, цветовая подсветка синтаксиса в листингах программ, размещение плавающих иллюстраций и т.п. выполняются автоматически  $\text{\TeX}$ -процессором в пакетном режиме, и на выходе получается красивый печатный или электронный (.pdf) документ.

Единственная область, не удобная в  $\text{\LaTeX}$ -верстке — создание сложных таблиц. Для этого были созданы визуальные редакторы, позволяющие отрисовать структуру таблицы мышью, а затем заполнить готовый шаблон данными.

## 107.10 Установка MikTeX под ☐Windows

## 107.11 Структура документа

### 107.11.1 Заголовочный файл или блок

### 107.11.2 Стили документа

### 107.11.3 Пакеты

### 107.11.4 Автор и название

### 107.11.5 Верстка титульных страниц

### 107.11.6 Оглавление

## 107.12 Верстка слайдов

## 107.13 Список литературы и цитирование

$\text{\LaTeX}$  умеет мощную подсистему управления цитированием и списками литературы. В простейшем случае, например при написании единственной статьи, раздел *библиографии* можно создать в том же документе, добавив в конец thebibliography:

```
\documentclass{article}
```

```
\input{header}
```

```
\author{Вася Пупкин}
```

```
\title{Пример статьи с цитатами}
```

```
\begin{document}
```

```
\maketitle
```

В статье используются книги: \cite{A} и \cite{B}

```
\begin{thebibliography}{99}
```

```
\bibitem{A} Книга А
```

```
\bibitem{B} Книга В
```

```
\end{thebibliography}
```

```
\end{document}
```

Но если вы регулярно работаете с документацией, или часто пишете статьи, возникает естественное желание вынести весь список литературы в отдельную базу данных, прописать авторов, названия, издательства и т.п. Это делается с помощью программы **biber** и пакета **biblatex**.

Пример использования этой системы вы легко найдете в исходниках этой книги:

- файл **header.tex** содержит секцию подключения пакета и подгрузки библиофайлов:

```
% books bib management
\usepackage{biblatex}
\addbibresource{../bib/python.bib}
\addbibresource{../bib/eskd.bib}
...
```

- библиофайлы хранятся в соседнем репозитории `../bib`, скопированном с <https://github.com/ponyatov/bib>.
- порядок вызова `pdflatex` и `biber` см. `Makefile`



107.14 Команды секционирования: часть, глава, раздел,...

107.15 Таблицы

107.16 Формулы

107.17 Перекрестные ссылки и гипессылки

107.18 Листинги скриптов и текстовых данных

107.19 Подготовка иллюстраций

Подготовка иллюстраций — одна из самых геморных тех в создании документации, и ее верстке для бумажных и электронных изданий.

Предпочтение нужно отдавать векторным форматам, за исключением фотоиллюстраций. В идеале скриншоты также хорошо бы переводить в векторный формат, но пока инструмент для этого не найден, поэтому выходные файлы будут пухнуть в объеме.

Для подготовки векторных иллюстраций: схем, графиков, диаграмм и т.п. используйте пакеты, принимающие на вход программы на специализированном языке программирования, легко читаемым человеком. В этом случае у вас сохранится отслеживать изменения, читая логи [VCS](#).

Обратите внимание на возможность использования стилевых файлов на весь проект (для всех иллюстраций в книге например). Их использование даст профессиональный вид продукту, при этом со-

храниться возможность взять и переформатировать 100500 схем в 10-томнике, поменяв шрифт, цвета, толщины линий, зазоры между элементами и т.п.

Пользуйтесь только относительными единицами размеров, и привязывайтесь к размерам шрифтов, это даст возможность использовать готовую иллюстрацию в нескольких проектах с разными размерами бумаги и наборами используемых шрифтов.

### 107.19.1 Графики GNUPLOT

Самый простой способ получить график простой аналитической функции или экспериментальных данных — воспользоваться утилитой **GNUPLOT**.

Оценить возможности можно вот по этому **видео**

Примеры **на википедии**

Примеры выложенные вместе с текстом **на языке gnuplot**

### 107.19.2 Схемы и графы в GraphViz

Для отрисовки графов и схем, легко к ним сводящихся, можно использовать пакет **GraphViz** и язык **Dot**.

### 107.19.3 PGF/TikZ

Сложные графики можно рисовать с помощью пакета **PGF/TikZ**, но для его работы нужна установленная  $\text{\LaTeX}$ -система. Этот пакет предназначен прежде всего для набора и верстки изданий с множеством сложных схем.

### 107.19.4 GLE

GLE — универсальный язык опиания векторных графических объектов с элементами языка программирования. Поддерживает вычисления, типовые конструкции программирования (циклы, условия, рекурсию).

- графики
- 3D графики
- диаграммы
- фракталы
- электронные схемы
- исчо

## 107.20 Верстка электронных изданий

Для электронных изданий, предназначенных для чтения с различных экранов как компьютера, так и портативных устройств, существует ряд ограничений и рекомендаций, из-за особенностей экранов: малый размер, низкое разрешение, поддержка цвета (TFT vs e-Ink) и т.п.: [2]

Установка полей в .PDF:

```
\hypersetup{  
pdftitle={Азбука халтурщика-АРМатурщика},  
pdfauthor={ruOpenWrt, HackSpace Чебураторный завод, Bill Collis (part 1)}  
}
```

## Часть XIII

### Куча

В этот раздел собраны все материалы, не вошедшие в основную часть потому что слишком сложны для начинающих, не попадают не в один раздел по тематике, или не вписались по каким-то другим параметрам.

Все новые материалы также сначала попадают сюда, а потом принимается решение об их переносе в основную часть.

Часто сюда пишут статьи те, кто принимает участие в создании книги эпизодически, или те, у кого нет достаточно времени заниматься их подготовкой.

# Список литературы

- [1] Bill Collis. *An Introduction to Practical Electronics, Microcontrollers and Software Design*. 2-е изд. 2014. URL: <http://www.techideas.co.nz/>.
- [2] Alan Wetmore. *e-Readers and L<sup>A</sup>T<sub>E</sub>X*. URL: <https://www.tug.org/TUGboat/tb32-3/tb102wetmore.pdf>.
- [3] Аллен Дауни. *Думать на языке Python: Думать как компьютерный специалист*. 1999. URL: <https://drive.google.com/file/d/0B0u4WeMj0894Q2hWV1Qw0FFQOVk/view?usp=sharing>.
- [4] Г. Россум и др. *Язык программирования Python*. Stichting Mathematisch Centrum, 1990–1995 и др., 2001, с. 454. URL: <http://rus-linux.net/MyLDP/BOOKS/python.pdf>.