Author	Dmytro Nikandrov
License	

Практика 1. Программируем ATMEL AVR используя свободные инструменты на GNU/Linux

Первые шаги
Для начала вам потребуется компьютер с установленной системой Linux: Debian или Ubuntu. И плата с любым микроконтроллером AVR, в данном примере мы будем работать с Arduino.

Скачайте последнюю версию среды разработки Eclipse Kepler учитывая тип вашей ОС (дистрибутив, разрядность).
Установите утилиту для прошивки микроконтроллера avrdude и библиотеки для работы с AVR, для этого в приложении Терминал введите:
```
$ sudo apt-get install gcc-avr avr-libc avrdude
```
Чтобы прошивать чип, программе надо получить доступ на работу с последовательными портами. Изначально это запрещено. В терминале наберите:
```
$ whoami
```
(как результат выдаст ваш username)
Давайте проверим что вы состоите в группе “dialout”:
```
$ groups username
```
Если в выданном результате нужной нам группы нет, набираем:
```
$ a='username'; sudo usermod -aG dialout $a
```
и после ввода пароля, вас добавят в группу “dialout”, которая имеет неограниченный доступ ко всем последовательным портам в Linux, включая виртуальные,
которые иногда применяются для устройств подключаемых через USB.

Запустите Eclipse и:

Установите CDT Plugin (C/C++ Development Tools):
- Help -> Install New Software…
- Work with: (your current Eclipse version) то есть Luna
- Выберите “Programming Languages” и там “C/C++ Development Tools”
- Согласитесь и выполните перезапуск Eclipse.

Установите AVR Eclipse Plugin:
- Help -> Install New Software…
- Добавьте новый репозиторий: http://avr-eclipse.sourceforge.net/updatesite/
- Скачайте AVR Eclipse Plugin 2.4.1 и выполните перезапуск Eclipse.

Настало время подключить вашу Arduino к компьютеру.
Подключите её обычным способом к USB.
Сразу наберите в терминале:
```
$ dmesg | tail
```
Вам надо найти слово "**ttyACM0**" или похожее, это имя порта к которому подключилась ваша

плата и оно вам пригодится далее.

Создайте новый проект с любым именем, например “AVR_blink_led”:

- Идите в меню File > New > Project... > C/C++ > C Project
- Project Type: раскройте AVR Cross Target Application, выберите Empty Project и AVR-GCC Toolchain, Нажмите next…
- Снимите галочку “Debug” (в режиме отладки, hex файл прошивки не генерируется и avrdude не сможет залить её в чип)
- Нажмите Advanced settings… AVR -> AVRDude -> Programmer configuration…

Теперь если у вас

Arduino Uno Создайте new programmer и назовите его “Arduino Uno”. Убедитесь что вновь созданная вами конфигурация программатора выбрана для текущего проекта. • Programmer Hardware: Arduino • Override default port: /dev/ttyACM0 or similar • Override default baudrate: 115200 AVR -> Target Hardware: • MCU Type: ATmega328P (or load from MCU) • MCU Clock Frequency: 16000000 (значение по-умолчанию генератора тактов Arduino Uno)	Arduino Mega или MegaADK Создайте new programmer и назовите его “Arduino Mega”. Убедитесь что вновь созданная вами конфигурация программатора выбрана для текущего проекта. • Programmer Hardware: Atmel STK500 Version 2.x firmware • Override default port: /dev/ttyACM0 or similar • Override default baudrate: 115200 AVR -> Target Hardware: • MCU Type: ATmega2560 (or load from MCU) • MCU Clock Frequency: 16000000 (значение по-умолчанию генератора тактов Arduino Mega/MegaADK)

Нажмите Apply и OK чтобы закрыть окно свойств и нажмите Finish чтобы создать новый проект.
На вопрос "Open Associated Perspective" дайте ответ Yes.

Создайте новый файл исходных кодов:
Идите в меню File > New > Source File
введите в строку "Source File:" имя "main.c"
скопируйте в него этот код:

```
#include <stdio.h>
#include <avr/io.h>
#include <util/delay.h>

#define LED PB7 // LED is on digital pin 13 or pin 7 of AVR's Port B

void initIO(void)
{
     DDRB |= (1<<LED);
}

int main(void)
```

```
{
	initIO();
	while (1)
	{
		PORTB |= (1<<LED); // set
		_delay_ms(500);
		PORTB &= ~(1<<LED); // clear
		_delay_ms(500);
	}
	return 0; // never reached
}
```

Примечание: Если вы используете Arduino Uno то в коде выше вам следует переопределить порт LED с PB7 на PB5,
так как светодиод на плате Arduino Uno подключен к Port B5.

Не забудьте сохранить main.c перед тем как продолжать (File -> Save).
Соберите проект Project -> Build Project
В оснастке Console посмотрите лог сборки проекта. Такие параметры как -mmcu= и -DF_CPU= должны соответствовать вашей плате.
Если в первом параметре указан не ваш микроконтроллер, а во втором неправильная его рабочая частота, тогда вы поймали ошибку в Eclipse :)
Для того чтобы исправить это пройдите в свойства проекта и повторите шаги по настройке AVRDude и AVR Target Hardware описанные выше.

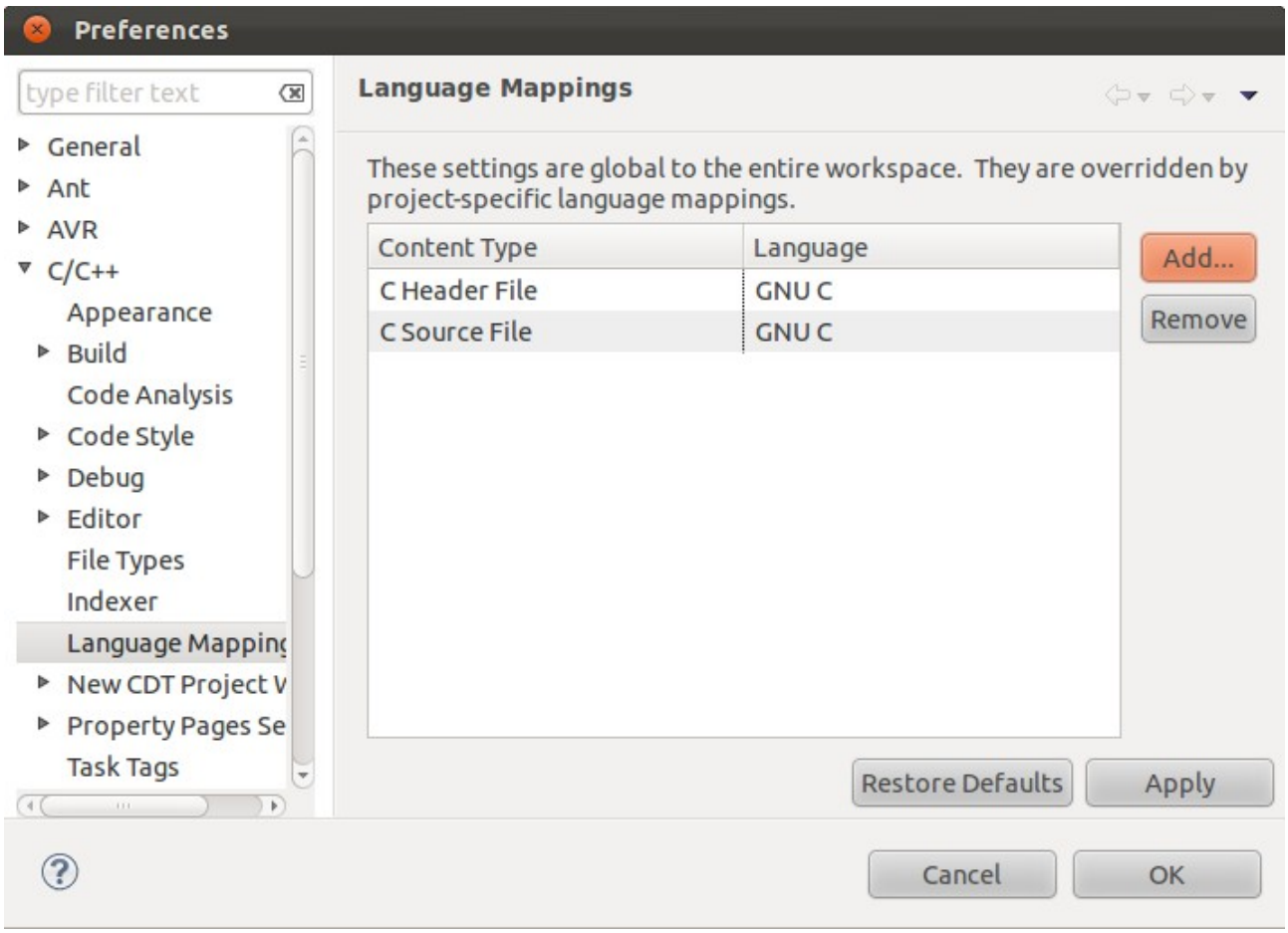
Нажмите на AVR Button в среде Eclipse для прошивки чипа сгенерированным hex файлом из проекта AVR_blink/Release/AVR_blink.hex.
Светодиод вашей Arduino должен быстро мигать. Если почему-то он не заработал, правым щелчком мышки выберите Properties вашего проекта. Проверьте что настройки AVR и Programmer установлены так как описанно выше.

Вот и всё! Надеемся что проект у вас успешно собрался и светодиод на ардуинке неистово мигает :)

P.S. В некоторых версиях Eclipse, определённые мнемоники AVR такие как DDRB (data direction register port b) не распознаются.
Чтобы решить эту проблему идём в Preferences:

C/C++
Language Mappings
Добавьте такие соответствия:
Content Type: C Header File / Language: GNU C
Content Type: C Source File / Language: GNU C
Не забудьте сразу нажать Apply, а также после этого перезапустить Eclipse.

Ваше окно настроек Language Mappings должно выглядеть так как на этом снимке экрана:



Если по-прежнему не работает, попробуйте добавить эту строку в начало вашего исходного файла main.c:

```
#include <avr/iom128.h>
```