Обмен опытом и передовые идеи по использованию свободного ПО в образовании

# Свободные ПО и электроника


Что общего у среды программирования для детей *Scratch* и свободной вычислительной платформы *Arduino*? **Александр Казанцев** даст ответ, сделав управление аппаратными устройствами доступным школьнику.

**Наш эксперт**

**Александр Казанцев**
К. т. н., доцент кафедры информатики Глазовского государственного пединститута, руководитель проекта EduMandriva, автор локализаций (и разработок) для Mandriva, *Klavaro*, *Gambas2* и *LXDE*.

Программирование – увлекательная вещь, но результат своего труда можно увидеть лишь на экране. То, что можно потрогать физически, вызывает больший интерес, чем виртуальная копия. Изучение алгоритмов дает понятие «исполнителя», что в большинстве детских умов ассоциируется с роботами или хотя бы электронными устройствами. Программируя реальное оборудование и взаимодействуя с ним, учащийся лучше понимает, для чего в принципе нужна программа и как работают вычислительные системы в целом. Наконец, когда какая-то «железка» вдруг выполняет твои команды, или персонаж на экране начинает слушаться только что собранной электронной схемы – это просто интересно и познавательно.

И здесь нам помогут свободные ПО и электроника. Проект Arduino (см. врезку справа) позволит войти в этот мир с минимальными затратами. Единственное «но» – программы (т. н. «скетчи») для используемых в Arduino микропроцессоров (ATMega) пишутся на ассемблере или с использованием специальных трансляторов с других языков. Это уровень студентов вуза, причем продвинутого. Среда *Arduino IDE* представляет верх минимализма и аскетизма и требует знания языков уровня C или Java. Как применять это в школе?

Спешу вас успокоить – мы будем использовать плату Arduino вместе со *Scratch* (см. врезку внизу). *Scratch* – это среда программирования «из кирпичиков», основанная на Logo, и с ней может справиться ребенок, начиная с 10–12 лет (при эмуляции ScratchBoard) или 14–15 лет – в случае *Catenary*.

## Исходные материалы

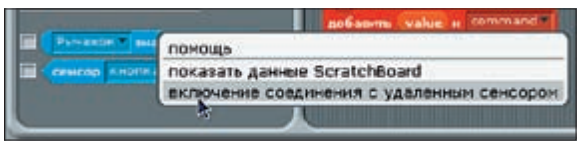
Итак, нам понадобятся: среда программирования *Scratch* версии не ниже 1.3 (мы возьмем 1.4.0), плата *Arduino* (подойдет любая совместимая; у нас была Arduino Duemilanove), светодиоды трех цветов, кнопка, резисторы на 1 кОм (есть в любом магазине радиотоваров) и провода. Не помешают макетная плата или паяльник, хотя можно собрать все, просто скрутив детали (конечно, это не так красиво и надежно). Разумеется, нужен дистрибутив Linux. Вы можете повторить эти действия и под Windows, но разбираться придется самим.

Первым делом установим *Scratch*. Для этого воспользуйтесь менеджером пакетов вашего дистрибутива или скачайте последнюю версию с сайта проекта (правда, найти ее там непросто). Установка и настройка ПО для Arduino освещены в номерах **LXF**, упомянутых во врезке, и этого мы касаться не будем. Не забудьте приготовить саму плату и будьте осторожны при обращении с электроникой – не касайтесь ее голыми руками. Вам это не повредит, но детали можно «убить» статическими электричеством.

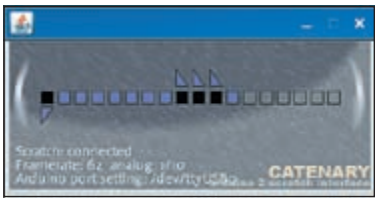
## Управляем из Scratch

Мы не станем заниматься «настоящим» программированием – взамен, будем управлять платой Arduino напрямую из *Scratch*, не загружая в нее скетчи. И в этом нам поможет проект *Catenary* (http://scratchconnections.wik.is/User:Chalkmarrow/Catenary). Распакуйте архив с программой в вашу домашнюю директорию и действуйте по приведенной инструкции.

Сперва подключите Arduino к компьютеру и залейте в нее код, превращающий ее в плату ввода/вывода. Запустите *Arduino IDE* (обычно командой **arduino**) и убедитесь, что в меню Tools > Board и Tools > SerialPort выбраны правильная плата и порт. Далее, зайдите в File > Examples > Firmata и выберите StandartFirmata. После того, как скетч откроется, залейте его на плату с помощью File > Upload I/O или кнопки на панели меню. Плата готова к управлению. Не забудьте подключить светодиод к цифровому контакту 13 (одну ножку в GND, другую в DIGITAL 13).




› **Не забудьте подключить сенсоры!**




**› Catenary отображает пользователю информацию в таком «минималистичном» виде.**

Запустите *Scratch*, откройте в нем проект, идущий в архиве с программой – **Scratch_Catenary1(Blink).sb**, и (ни в коем случае не запуская его) включите соединение с удаленным сенсором, выбрав синюю вкладку Сенсоры и кликнув правой кнопкой на любом блоке со словом «сенсор».

Затем запустите *Catenary* (желательно от имени root, чтобы не иметь проблем с доступом к устройствам, и обязательно после *Scratch*). Перейдите в директорию **Catenary/application.linux/** и скомандуйте:

```
chmod +x ./Catenary
sudo ./Catenary
```

В моей системе (EduMandriva) *sudo* не настроен, и нужно использовать *gksu* или *kdesu*.

```
gksu ./Catenary
```

Если ваша система – 64-битная, *Catenary* откажется работать, сославшись на 32-битные библиотеки. Чтобы исправить это, создайте символьную ссылку на 64-битную библиотеку **librxtxSerial.so** из **/usr/lib64/rxtx** или **/usr/lib64/rxtx-x.y.z**. Находясь в одном каталоге с запускаемым файлом, наберите (**x.y.z** нужно заменить на актуальную версию):

```
rm ./librxtxSerial.so
ln -s /usr/lib64/rxtx/librxtxSerial.so.x.y.z ./librxtxSerial.so
```

Кроме того, нужно скомандовать:

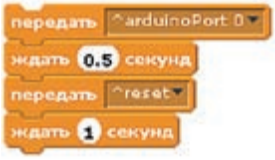
```
rm ./lib/RXTXcomm.jar
ln -s /usr/share/java/RXTXcomm.jar ./lib/RXTXcomm.jar
```

В процессе написания статьи я обнаружил, что данные передавались на плату, но не влияли на состояние выходов. Несколько часов поисков показали, что дело еще в одной библиотеке. Возьмите файл **processing-arduino-0017.zip** с LXFDVD и замените библиотеку в **/Catenary/application.linux/lib** на содержащуюся в архиве (**/arduino/library/Arduino.jar**), не забыв назвать ее с маленькой буквы. Это может и не потребоваться; но предупрежден – значит, вооружен.

Наконец, появится окно, изображенное выше. Первый слева голубой квадрат – цифровой контакт 2, последний – 13. Серыми квадратиками обозначены аналоговые контакты 0–5 (в нумерации *Scratch* – 14–19).

Теперь вернитесь к *Scratch* и попробуйте запустить проект. Если все пойдет нормально, вы увидите мигающий светодиод.

Разберем, как устроена программа. Второй персонаж (Catenary) нужен для обеспечения работы переменных, и трогать его не нужно. Программа пишется в спрайте «кота». Все передаваемые команды начинаются с **^**. Сперва идут блоки инициализации платы – задания порта (**^arduinoPort**) и сброса состояния (**^reset**).




Далее мы переключаем цифровой контакт 13 на вывод (**^pinMode 13 output**).


## Подробнее о Scratch

Если вы ранее не сталкивались со *Scratch*, вам стоит познакомиться с ним поближе. Это новая среда программирования, которая позволяет детям создавать собственные анимированные и интерактивные истории, игры и другие произведения. *Scratch* базируется на традициях языка Лого и написан на Squeak. В среде *Scratch* используется метафора кирпичиков Лего, собирать из которых простейшие конструкции могут даже самые маленькие дети. Но, начав с малого, можно развивать и расширять свое умение строить и программировать. *Scratch* создавался специально для того, чтобы подростки 10–16 лет использовали его самостоятельно в сети внешкольного обучения.

На самом деле это больше, чем среда программирования – это инструмент создания интерактивных мультфильмов, музыки, игр, историй, которыми можно поделиться с другими. Дети могут завести себе учетную запись на сайте проекта http://scratch.mit.edu/ и получать/давать доступ к своим проектам из любой точки земного шара. Наличие специального модуля позволяет встраивать проекты *Scratch* в любой web-сайт (требуется Java), поэтому он может в какой-то мере служить заменой Flash. Вы можете найти больше информации по *Scratch* на таких сайтах, как http://scratched.media.mit.edu, http://letopisi.ru, http://ru-scarystories.blogspot.com.

После этого начинаем мигать светодиодом, переводя состояние цифрового канала из выключенного (**low**) в включенное (**high**).


Ниже приведена модификация программы, передающая азбукой Морзе сигнал SOS (три коротких, три длинных, три коротких). Чтобы добавить новый текст в блок Передать, щелкните на стрелочке сбоку, выберите Новый, и введите нужный текст.


## Скорая помощь

После нажатия кнопки сброса на плате и других подобных действий нужно действовать по алгоритму:

1. Подключить Arduino и залить прошивку.
2. Запустить *Scratch*, включить связь с сенсорами и написать программу.
3. Запустить *Catenary*.
4. Запустить программу в *Scratch*.

»

## Arduino и его родственники

**LXF** уделял Arduino немало внимания (см. номера 100–101 и 103–106 на wiki.linuxformat.ru), поэтому мы остановимся на том, где найти эти устройства в России и как получить помощь на родном языке. Оригинальные платы Arduino появляются (по мере поступления) в интернет-магазине ГНУ/Линуксцентра (www.linuxcenter.ru). Существуют два других проекта, выпускающих свои Arduino-совместимые платы: Robocraft (http://robocraft.ru/) и Freeduino (http://freeduino.ru/). На их сайтах проектов можно найти много полезной информации по работе с платой.

## Не только выход

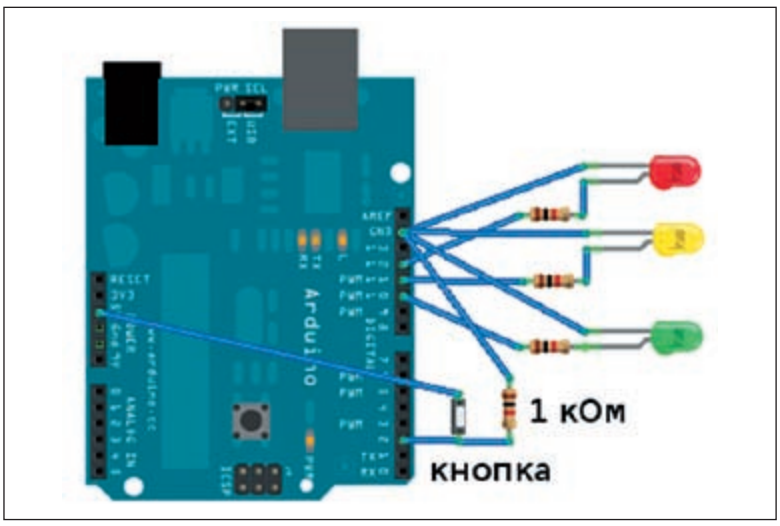
Но *Catenary* позволяет не только управлять цифровыми выходами – вы также можете изменять состояние аналоговых контактов и получать информацию с цифровых и аналоговых входов, то есть подключать внешние датчики.

Попробуем смоделировать двухрежимный светофор: изначально он будет работать как обычный (красный/желтый/зеленый), а по нажатию кнопки – переходить в режим мигающего желтого и обратно.

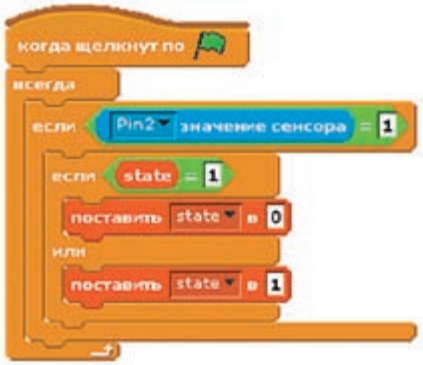
Соберите схему, как показано на рисунке (резисторы нужны для устранения дребезга контактов и ограничения тока на светодиодах). После этого создайте нижеследующий программный код, взяв за основу предыдущий проект. Помните, что персонаж *Catenary* жизненно необходим, и удалять его нельзя.

Разберем алгоритм, который мы применили при создании модели. После инициализации задаются режимы (**^pinMode**) контактов: мы будем использовать цифровые контакты 10, 11 и 12 для управления светодиодами (зеленым, желтым и красным, соответственно) и цифровой контакт под номером 2 как вход кнопки.

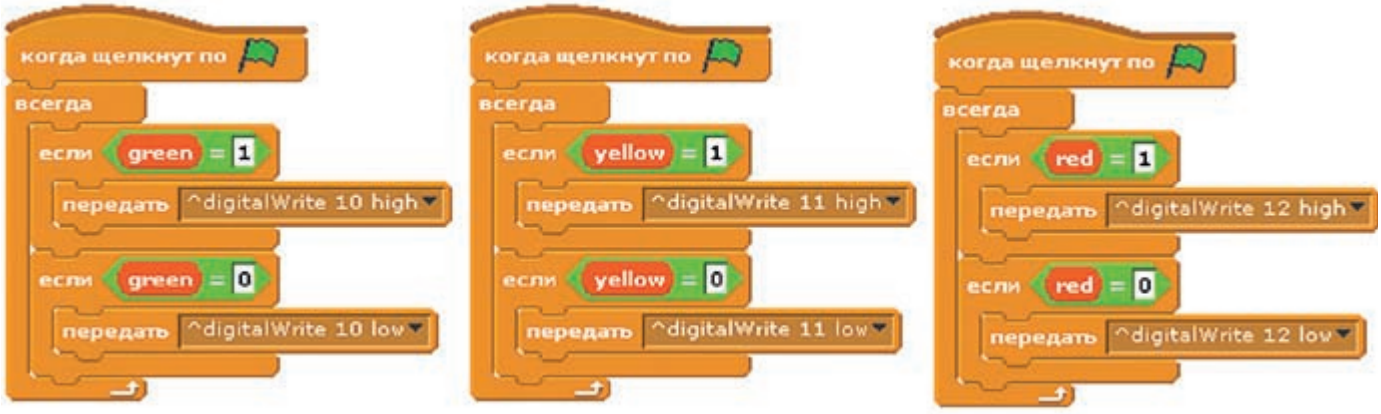

**› Принципиальная схема нашего светофора.**




Считывание данных с кнопки происходит через сенсор **Pin2**. Если его нет в выпадающем списке сенсоров (синяя вкладка Сенсоры), запустите только что набранный код. Как только сенсор фиксирует нажатие, мы изменяем состояние **state** на противоположное.




После старта мы входим в бесконечный цикл, в котором, в зависимости от значения переменной **state**, реализуем или дневной режим светофора (**state = 0**) или ночной («мигалка» – **state = 1**).


Три переменных (**green**, **red** и **yellow**) определяют, включен ли светодиод соответствующего цвета. Три блока, в зависимости от значения переменных, передают на плату соответствующие уровни для цифровых контактов (**high** и **low**).

Вы можете красиво оформить данную модель, сделав реальный светофор, а также доработать программу, обеспечив правильное переключение света – то есть добавив возможность одновременного включения красного и желтого перед зеленым и наоборот. Также можно поэкспериментировать и с задержками сигналов светофора.

## Плата для Scratch

Это все, конечно, интересно, но сложновато для школьников 5–6 класса. А можно ли придумать что-то подобное и для них? Да, если потрудиться и собрать на основе Arduino так называемый ScratchBoard. В оригинале это плата PicoBoard, разработанная специально для использования со *Scratch* и имеющая в нем встроенную поддержку со стороны блоков программирования (http://info.scratch.mit.edu/Sensor_Boards). Проблема в том, что ScratchBoard нужно заказывать из-за рубежа (хотя цена его – не более $50); но с помощью Arduino мы сделаем подобную плату сами. По адресу http://scratch.mit.edu/forums/viewtopic.php?id=28188 можно найти различные варианты реализации и советы. Воспользуемся одним из них: http://www.yengawa.com/scratch_arduino.

> «Для школьников 5—6 класса можно собрать ScratchBoard.»

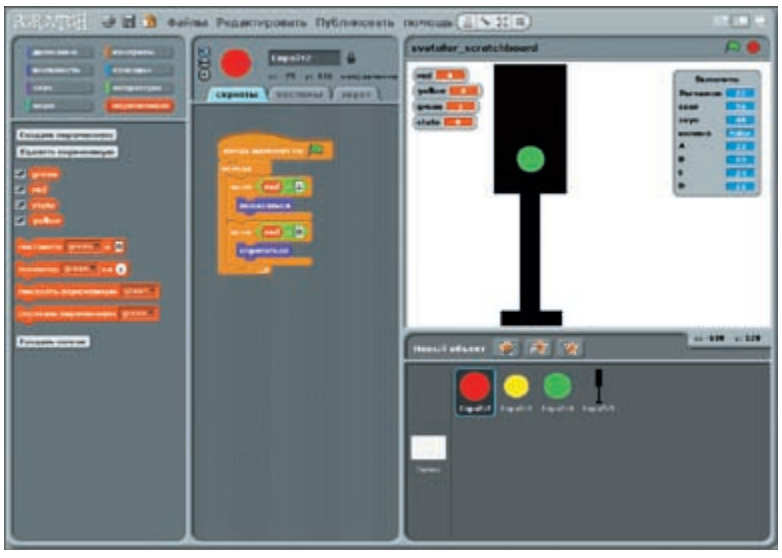
Для начала, реализуем не все функции (это тема для отдельной статьи – напишите нам на letters@linuxformat.ru, если такой материал представляет интерес). Мы воспользуемся кнопкой от предыдущего проекта. Позже можно будет добавить датчики освещения, звука, ползунковый резистор и четыре резистивных входа.

Собрав все вместе и подключив к плате, нужно запустить *Arduino IDE* и залить в устройство прошивку эмулятора, которую можно найти по адресу http://www.yengawa.com/sites/default/files/uploads/ScratchBoard.pde или на **LXF**DVD.

Далее, запустите *Scratch* и, щелкнув правой кнопкой мыши по блоку Значение сенсора в синей вкладке Сенсоры, выберите Показать данные ScratchBoard. Затем щелкните правой кнопкой по блоку, возникшему в окне отображения, и выберите порт, к которому подключена ваша плата. Теперь можете проверить, изменяются ли значения сенсоров при взаимодействии с ними. Если все работает, то можно приступать к программированию (блок можно просто скрыть). У вас может возникнуть проблема с доступом к порту, поэтому если после подключения платы ничего не поменялось, дайте в терминале от имени root команду

```
chown user /dev/ttyUSB0
```

где **user** – ваш пользователь, а **/dev/ttyUSB0** – порт, к которому подключена плата Arduino.



**› Наш светофор не только умеет мигать в двух режимах, но и позволяет переключать их кнопкой на нашей импровизированной ScrAtchrDuinoBoard.**

Теперь можно протестировать нашу видоизмененную программу модели светофора, но уже с использованием возможностей нашей «ScratchBoard» и без реальных светодиодов. Графический текст программы вы можете найти на **LXF**DVD (файл **ScratchBoard_svetofor.pdf**) или (наряду с другими программами) по адресу http://wiki.edumandriva.ru/wiki/index.php/Arduino.

Мы рассмотрели только самые основные моменты использования связки Scratch–Arduino. «За бортом» остались управление двигателями, использование сенсоров, снятие показаний с датчиков и другие не менее интересные вещи. Напишите нам или оставьте сообщение на форуме forum.linuxformat.ru, если данная тема вам интересна и ее стоит развивать. LXF