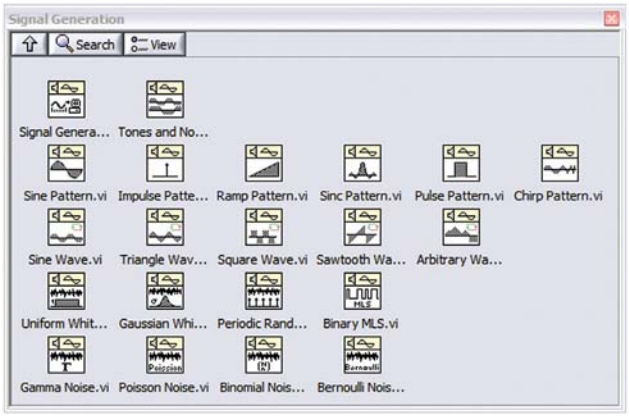
Учебный практикум по LabVIEW

NI LabVIEW™ Toolkit for LEGO® Mindstorms® NXT, частично рассмотренный в ПиКАД №3-2007, не мог не заинтересовать "детей всех возрастов". Тем более, что многие в Украине уже стали счастливыми обладателями образовательного конструктора LEGO® Mindstorms® NXT. И число LEGO-манов растет с каждым днем и, не сомневайтесь, будет расти. Поэтому продолжим знакомство с возможностями этого Toolkit. Но сначала обязательная программа, а Новогодние сюрпризы – потом.

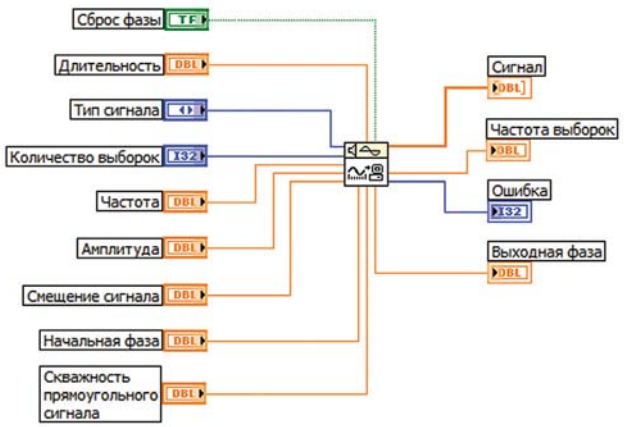
После появления в LabVIEW 7 концепции Express VI, и, в частности, функции ***Simulate Signal***, некоторые пользователи начали потихоньку забывать о существовании других блоков формирования сигналов. И напрасно. Конечно, к безусловным достоинствам *Simulate Signal* можно отнести его простоту и быстроту конфигурирования. Но, как известно, плюсов без минусов не бывает. Так вот, за эту простоту приходится платить низким быстродействием данной функции даже на серьезных компьютерах. Именно поэтому рекомендуется использовать ее в простых тестовых или не критичных по времени исполнения приложениях. В противном случае, во избежание недоразумений, воспользуйтесь более простыми для Вашего компьютера VI. О функциях генерации сигналов и шумов сегодня и пойдет речь ***(Functions ≫ Signal Processing ≫ Signal Generation)***:



Много в этой палитре интересных новых иконок VI. Самая первая из них - ***Signal Generator by Duration***. Этот генератор сигналов заданной длительности является многофункциональным и имеет большое количество входных параметров.

Первый вход ***Сброс фазы*** влияет на установку начальной фазы генерируемого сигнала. Если здесь установлено значение TRUE, то фаза сигнала устанавливается в соответствии со значением, указанным на входе ***Начальная фаза*** (по умолчанию установлено значение 0). Если же установить FALSE, то она будет равна значению выхода ***Выходная фаза***, которое было при последнем запуске данного VI. По умолчанию на этот вход

Signal Generator by Duration



подается значение TRUE. Далее будет рассмотрен примерчик, который более детально раскроет суть входа для сброса фазы.

На входе ***Длительность*** задается время в секундах равное длительности выходного сигнала. По умолчанию значение этого входа равно единице, а точнее - 1с.

При помощи регулятора ***Тип сигнала*** можно выбрать форму генерируемого сигнала:

- синусоидальный;
- косинусоидальный;
- треугольный;
- прямоугольный;
- пилообразный;
- линейно нарастающий;
- линейно спадающий.

Вход ***Количество выборок***, догадаться нетрудно, указывает на количество точек сигнала в выходном массиве. По умолчанию установлено 100 выборок.

Вход ***Частота*** определяет частоту выходного сигнала. Этот параметр обязательно нужно согласовать со значениями входов **Количество выборок** и **Длительность**, потому как в результате нарушения критерия Найквиста на выходе можно получить нечто похожее на шум. По умолчанию установлено значение 10 Гц.

Значение входа ***Амплитуда*** по умолчанию равно 1.

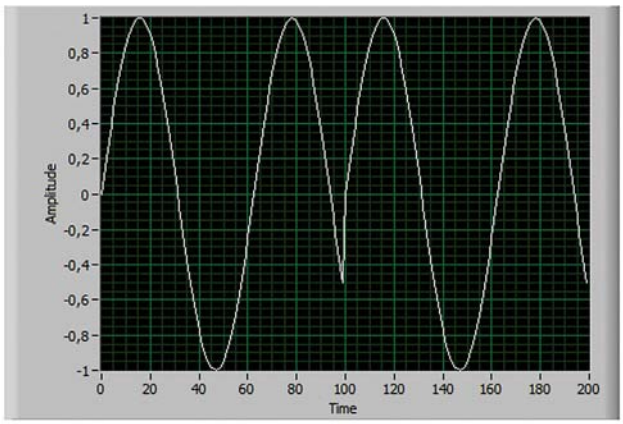
Вход ***Смещение сигнала*** задает постоянную составляющую выходного сигнала, т.е. указывает насколько уровень сигнала будет выше или ниже 0. По умолчанию смещение отсутствует.

Значение на входе ***Скважность прямоугольного сигнала*** задается в процентах. Немного странно. Потому как корректно ли значение в процентах называть скважностью? Этот параметр определяет время, в течение которого прямоугольный сигнал будет иметь высокий уровень. Параметр скважности применим только для прямоугольного сигнала, поэтому если выбрать другой тип сигнала, этот вход просто игнорируется. По умолчанию установлено значение 50 т.е. на выходе ***Сигнал*** Вы получите меандр-импульс, с одинаковой длительностью высокого и низкого уровня.

Выход ***Частота*** выборок указывает частоту дискретизации выходного сигнала, определяемую отношением количества выборок к заданной длительности.

А теперь рассмотрим небольшой пример, из которого станет ясно, каким образом работает вход ***Сброс фазы*** и относящиеся к нему начальная и выходная фазы. Для этого расположите рассматриваемый VI на поле блока диаграмм и укажите следующие входные параметры: тип генерируемого сигнала - синусоидальный, длительность - 0,8; частота - 2. Ну, а остальные параметры оставляем по умолчанию.

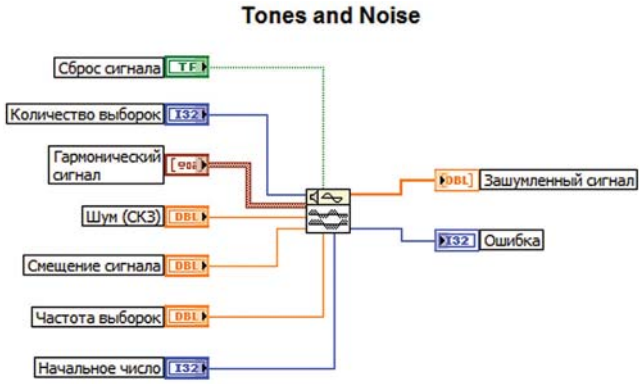
К выходу ***Сигнал*** подключаем ***Waveform Chart*** с максимальным значением по оси X равным 200, берем это все в структуру ***While Loop*** и не забываем поставить задержку секунды примерно на 2, чтобы успеть что-то увидеть. Регулятор ***Сброс фазы*** - в положении TRUE. Запускаем программу и, немного подождав, видим, что выходные массивы данных не привязаны друг к другу, и сигнал постоянно "срывается". Это происходит потому, что генерация каждого "куска" синусоиды начинается со сброса фазы в ноль (а точнее в то значение, которое указано в ***Начальной фазе***):



Попробуйте изменить значение ***Сброс фазы*** на FALSE и запустите программу. Вы увидите, что срывов сигнала уже не наблюдается. Это объясняется тем, что начальная фаза сигнала всегда устанавливается равной выходной фазе при последней генерации, что и позволяет получать непрерывный сигнал.

С первым VI наконец-то закончили. Поехали дальше! Рассмотрим блок, позволяющий генерировать зашумленный гармонический сигнал ***Tones and Noise***. Это генератор колебания и шума.

При подаче на вход ***Сброс сигнала*** значения TRUE для каждого нового генерируемого массива данных происходит сброс параметров гармонического колебания (фаза) и шума (начальное число). По умолчанию



установлено значение FALSE, поэтому сброса параметров сигнала и шума не происходит.

Вход ***Гармонический сигнал*** представляет собой кластер и содержит в себе основные параметры гармонического сигнала:

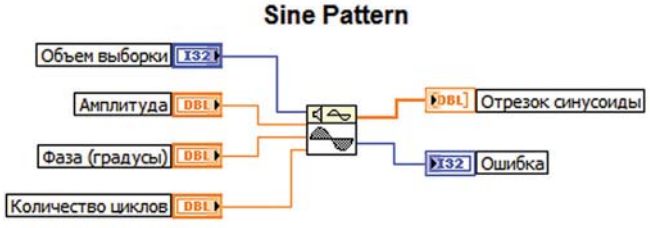
- частота (по умолчанию 10Гц);
- амплитуда (по умолчанию 1);
- начальная фаза (по умолчанию 0).

При помощи входа ***Шум (СКЗ)*** устанавливается среднеквадратическое значение Gauss-шума. По умолчанию на этом входе установлено значение 0, поэтому, если его не изменить, на выходе будет чистенькая синусоида.

Вход ***Частота выборок*** определяет количество выборок сигнала за секунду (частота дискретизации). По умолчанию установлено 1000 выборок/с.

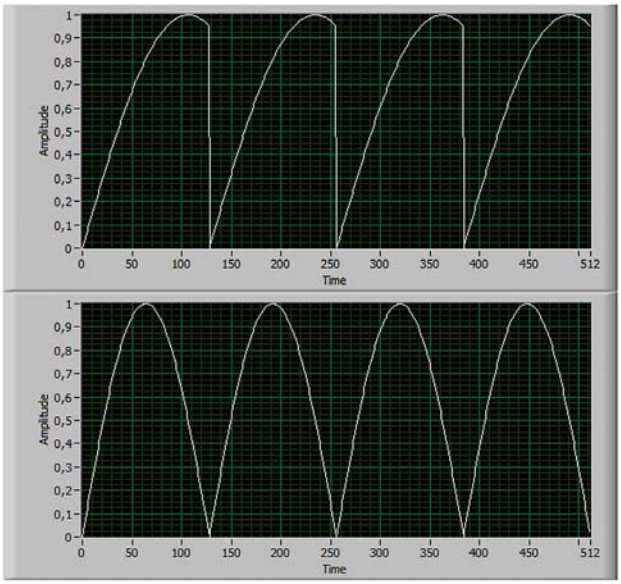
Вход ***Начальное число*** опрашивается только тогда, когда на входе ***Сброс сигнала*** установлено TRUE. В этом случае, если это число меньше либо равно нулю, VI для каждого массива использует вновь сгенерированную шумовую составляющую. Если же поставить единицу или более - будет все время повторяться ранее полученная шумовая составляющая. Попробуйте "поиграться" с этим регулятором самостоятельно, и тогда все станет понятней. Но долго не засиживайтесь **:)** - ведь Вас ожидают другие, не менее полезные функции.

Следующая группа VI используется для формирования детерминированных сигналов. Первым ее представителем является VI под названием ***Sine Pattern*** (генератор отрезка синусоиды):



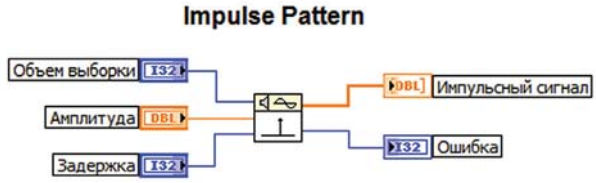
Не стоит останавливаться на входах ***Амплитуда, Фаза***, ***Объем выборки***, сразу перейдем к "сути" данного VI. Ключевым параметром является ***Количество циклов.*** По умолчанию установлен один цикл, то есть будет сгенерирован один период синусоиды. Но это число не обязательно должно быть целым, его можно задавать с точностью до нескольких знаков после запятой. Скажем прямо - в большинстве случаев ничего хорошего из этого не выйдет, но иногда это свойство может быть весьма

полезным. Ниже представлены два случая, когда количество циклов было задано дробным (в первом случае 0,3, во втором - 0,5):



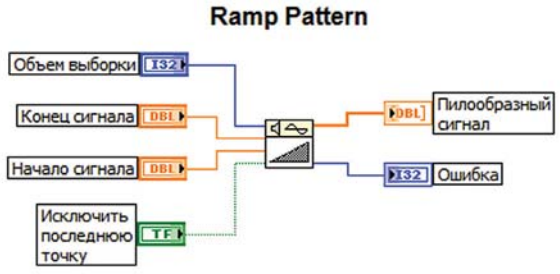
Как видите, в первом случае получился сигнал, вряд ли пригодный для чего либо, а во втором - положительный полупериод синусоиды, применяемый довольно-таки часто.

Далее в списке формирователей стоит генератор импульсного сигнала ***Impulse Pattern***. С его помощью Вы сможете получить импульс заданного уровня:



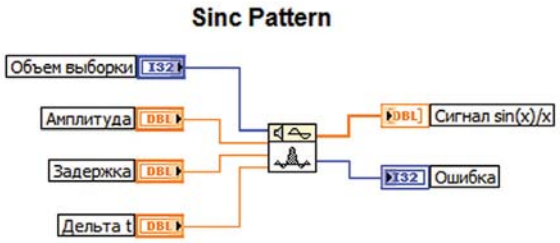
На вход ***Задержка*** фактически подается индекс элемента в выходном массиве, который будет отличным от нуля (задержка от начала генерируемого массива). Не пытайтесь подать на этот вход значение меньшее либо равное нулю - сразу получите сообщение об ошибке. Не верите - убедитесь сами.

После импульсного попробуем сформировать пилообразный сигнал при помощи ***VI Ramp Pattern***:



Входы ***Начало сигнала*** и ***Конец сигнала,*** как ни странно, определяют начальный и конечный уровни пилы. Установленные значения по умолчанию - 0 и 1, соответственно. Обратите внимание на то, что отсутствуют ограничения к этим входам, т.е. конечный уровень может быть ниже, чем начальный. Что Вы получите в таком случае? Конечно же, линейно-спадающий сигнал. Есть еще один интересный вход в этом VI - ***Исключить последнюю точку***. По умолчанию установлено FALSE. Если же установить TRUE, то последней точки выборки, соответствующей максимуму (минимуму), не будет в выходной последовательности.

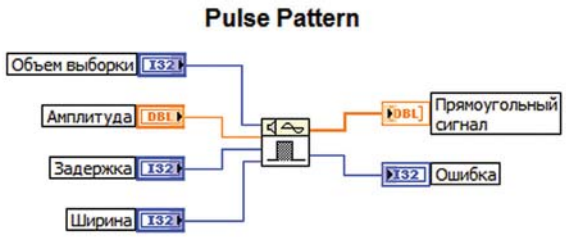
VI, генерирующий крайне полезный, например, в научных исследованиях, сигнал sin(x)/x, в LabVIEW именуется ***Sinc Pattern***:



Вход ***Задержка*** определяет место расположения центрального лепестка функции. Индекс максимального значения функции sin(x)/x определяется из соотношения:

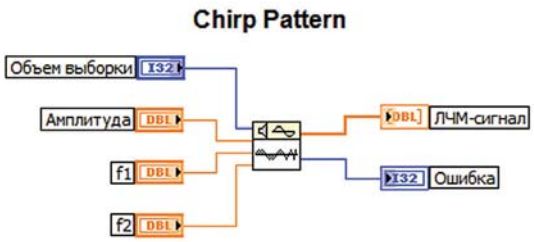
Индекс = задержка / Дельта t.

Значение ***Дельта t*** определяет частоту следования нулей функции. По умолчанию установлено число 0.1, что соответствует расстоянию между нулями равному 10 отсчетам (кроме случая с центральным лепестком, его ширина будет 20 отсчетов). Следовательно, если значение задержки установить равным 6,4, а остальные - по умолчанию, то мы получим генератор симметричной функции sin(x)/x.



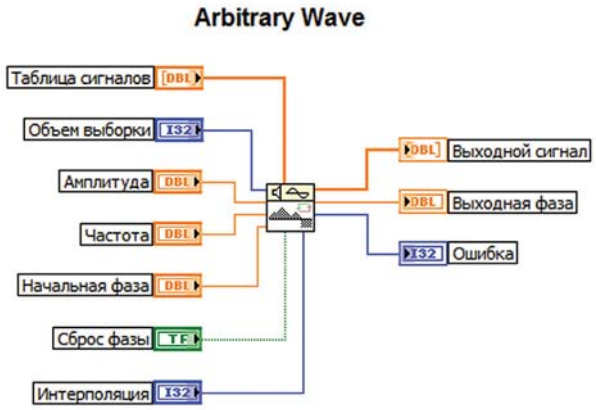
VI ***Pulse Pattern*** поможет Вам сгенерировать отрезок прямоугольного сигнала. Его вход ***Задержка*** определяет передний фронт сигнала (индекс первого элемента, имеющего высокий уровень), а ***Ширина*** - устанавливает количество отсчетов, для которых уровень сигнала будет высоким. Следует еще заметить, что сумма значений задержки и ширины не должна превышать количество отсчетов в выборке.

Если же Вам нужен сигнал с линейно-изменяющейся частотой (ЛЧМ), воспользуйтесь VI ***Chirp Pattern***:



Входы ***f1*** и ***f2*** определяют минимальное и максимальное значение частоты ЛМЧ-сигнала, т.е. значения, в пределах которых будет изменяться частота выходного сигнала. Эти значения задаются в нормированных единицах: отношение частоты к частоте дискретизации. Иными словами, они определяют, сколько точек будет приходиться на один период сигнала. Например: если установить значение 0.1, то за период будет получено 10 точек.

Далее в списке генераторов следуют ***Sine Wave***, ***Triangle Wave***, ***Square Wave***, ***Sawtooth Wave***, работа которых аналогична работе самого первого VI, рассмотренного сегодня. Эти специализированные VI могут генерировать только один тип сигнала. Поэтому их рассмотрение отнесем на самостоятельную проработку, и перейдем к функции генерации сигнала произвольной формы ***Arbitrary Wave:***

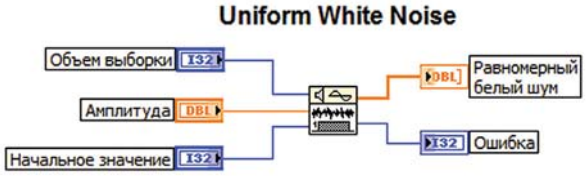


Вход ***Таблица сигнала*** является массивом, задающим форму колебания в пределах одного периода (весь сигнал указывается по точкам).

На вход ***Частота*** нужно подавать значение в нормированном виде подобно тому как и для ***Chirp Pattern***. Поэтому не удивляйтесь, что введя сюда значение большее единицы, на выходе получите прямую линию с нулевым уровнем.

Вход ***Интерполяция*** определяет вид интерполяции, применяемый при формировании сигнала. По умолчанию установлено значение "none", при этом никаких преобразований над данными со входа ***Таблица сигнала*** не выполняется. Если же поставить значение "linear", то VI будет выполнять линейную интерполяцию при формировании выходного массива.

Вроде как с основными сигналами все. Но есть в LabVIEW возможность генерировать еще и шумовые сигналы. Начнем с равномерного белого шума - ***Uniform White Noise:***

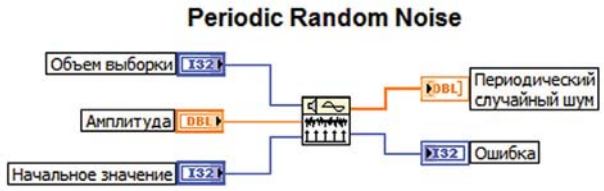


Этот VI генерирует псевдослучайный белый шум с равномерным законом распределения. Значение на входе ***Амплитуда*** указывает предельные как верхний, так и нижний уровни шума. По умолчанию амплитуда равна 0.

Следующим будет "шуметь" гауссовский белый шум ***Gaussian White Noise***:

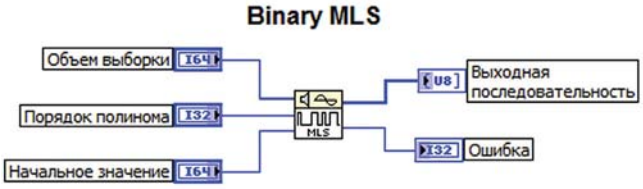
На выходе VI формирует псевдослучайную последовательность с нормальным распределением и заданным ***Стандартным отклонением***. По умолчанию параметр отклонения равняется единице.

Интересно звучит словосочетание "периодический случайный", не правда ли? Такое название получила функция ***Periodic Random Noise*** из-за того, что этот периодический случайный шум является результатом суммирования некоторого числа периодических сигналов:



В выходном массиве присутствуют все частоты, которые могут быть представлены целым числом периодов, исходя из установленного ***Объема выборки***. Амплитуды периодических сигналов указываются на соответствующем входе, а фазы выбираются случайным образом. Вот такой непростой шум получается.

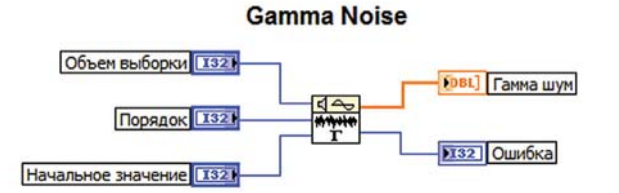
Двоичную последовательность максимальной длины формирует VI ***Binary MLS*** (maximum length sequence):

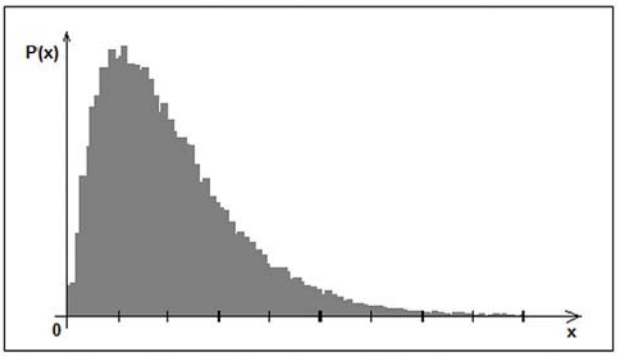


Возможно не каждому из Вас известно, что означает "двоичная последовательность максимальной длинны". Поэтому придется дать некоторые пояснения. Основные свойства этой последовательности следующие. Она является периодической с периодом $N = 2^n - 1$ выборок, где n - порядок полинома. Количество символов, принимающих значение единицы, на длине одного периода последовательности, равно 2^n-1, что на единицу больше, чем количество символов, принимающих значение ноль. ***Binary MLS*** - это еще цветочки, далее ягодки.

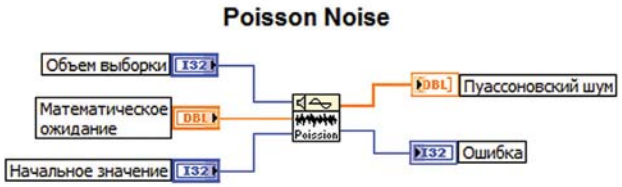
Шумовой сигнал на основе гамма распределения получил одноименное наименование - гамма-шум. Он реализуется с помощью VI ***Gamma Noise***.

Распределение шума при значении параметра ***Порядок*** равном 2 показано на графике. Чем выше будет число порядка, тем больше распределение шума будет напоминать нормальное.

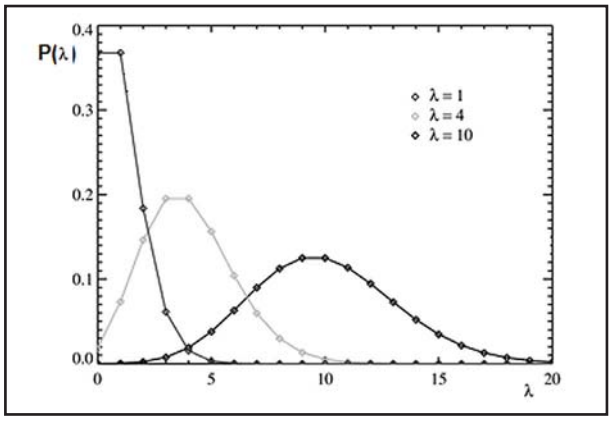




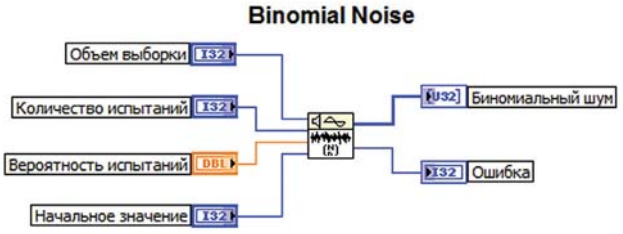
Нетрудно догадаться, что VI ***Poisson Noise*** генерирует шум на основе распределения Пуассона:



Представленные на графике несколько возможных вариантов распределений в зависимости от параметра ***Математическое ожидание*** (λ). Чем он больше, тем меньшим получится максимум распределения.

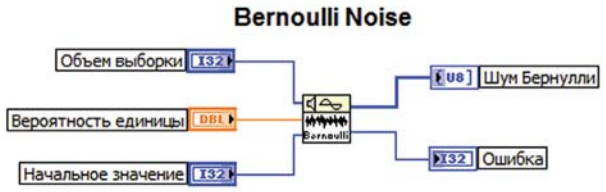


Биномиальный шум можно сформировать с помощью VI, который, и это естественно, именуется ***Binomial Noise***:



Функционирует этот VI следующим образом. Для каждого отсчета массива устанавливается ***Количество испытаний*** (по умолчанию 1); вероятность выпадения единицы (двоичная система) в каждом из этих испытаний равна значению, указанному на входе ***Вероятность испытаний***. После того, как для данного отсчета были получены все события, они суммируются. Например, если Вы установите количество испытаний равным 10, а вероятность 1 - на выходе будет прямая, имеющая уровень 10. Если вероятность будет 0,5 - получим кривую, среднее значение которой примерно равно 5 и т.д. Для того, чтобы это прочувствовать, не поленитесь и "поиграйте" с входами данного VI, тогда все сразу станет понятным.

Шум Бернулли генерирует VI ***Bernoulli Noise***:



В основе алгоритма получения шума лежит способ, эквивалентный подбрасыванию монеты с вероятностью того, что выпадет единица. Это в случае, если на входе ***Вероятность*** будет указано 0,5. Если же указать значение, скажем, 0.6, то игра в монетку будет не очень честной, потому как одна из граней будет выпадать с вероятностью 60%.

На этом обязательную программу текущего занятия можно считать выполненной. Вторую его часть посвятим, как и было обещано, NI LabVIEW™ Toolkit for LEGO® Mindstorms® NXT. А если конкретнее - созданию собственных функциональных блоков для младшего брата LabVIEW - LEGO® Mindstorms® NXT Software. И делать это следует в LabVIEW 7, потому как именно седьмая версия послужила основой среды разработки программного обеспечения для контроллера NXT.

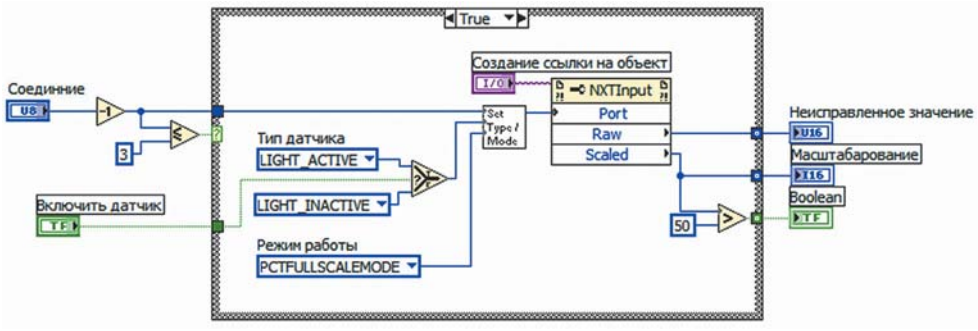
В качестве примера предлагается разработать функциональный блок для работы со световым датчиком. Первым делом, используя мастер создания нового NXT-блока, генерируется новый элемент. Для этого:

- выбираем ***Tools ≫ NXT Module ≫ New NXT Block Wizard***;
- вводим имя будущего блока в соответствующем окне, например ***Light.vi*** (Ваше название может быть каким угодно);
- выбираем ***Template Simple Sensor Block*** и нажимаем ***Create***, т.е. создаем шаблон;
- выбираем директорию, в которой будет сохранен Ваш новый функциональный блок, и нажимаем ***Current Folder***. При этом будут созданы все необходимые файлы поддержки будущего блока;
- жмем ***Close***, и сохраняем новые VI, если это требуется.

Шаг номер два - изменение встроенного Sub VI:

- в созданной директории находим файл ***Light.vi*** и открываем его;
- на поле блок-диаграммы делается двойной щелчок на ***Simple Sensor Sub VI***;

- модифицируем этот ***Sub VI*** следующим образом:
  - создаем булевский регулятор с названием "Включить датчик";
  - копируем константу ***Тип датчика*** и соединяем ее с Select.vi так, как показано на диаграмме;

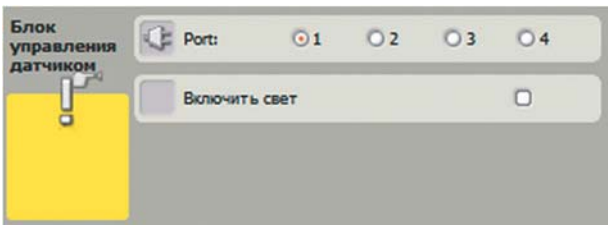


  - переходим на интерфейсную панель и подключаем созданный регулятор к схеме подключений connector pane, как вход;
  - сохраняем и закрываем этот Sub VI, после чего возвращаемся к Вашему ***Light.vi***;
- снова создаем регулятор ***Включить датчик*** и подключаем его к соответственному входу ***Simple Sensor Sub VI***;
- подключаем этот регулятор к схеме подключений - теперь он будет доступен при дальнейшей разработке Вашего элемента;
- не забываем сохранить модифицированный VI.

Следует отметить, что LEGO® Mindstorms® NXT Software поддерживает только числовой, булевский и текстовый типы данных. Во избежание недоразумений, используйте в своих проектах только названные типы данных.

Шаг третий - изменение конфигурации блока. Для этого необходимо открыть файл ***Config Light.vi*** и проделать буквально следующее:

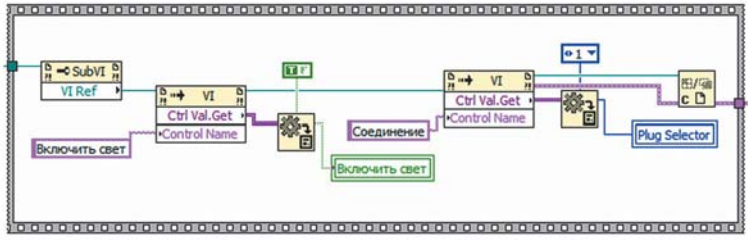
- в верхнем левом углу передней панели изменяется стандартный текст на тот, который будет описывать создаваемый блок, например, блок управления световым датчиком;
- конфигурируется внешний вид функциональной панели блока, с учетом добавленного переключателя включения/выключения датчика. Для этого рекомендуется использовать такие элементы: ***NXTToolkit >> NXT Controls >> Config Decorations >> One Row Background*** (декорация) и ***NXTToolkit >> NXT Controls >> NXT Checkbox*** (булевский регулятор с названием "Включить свет"). Остальные служебные индикаторы и регуляторы можно либо сместить вниз передней панели, либо сделать их невидимыми *(Hide)*. В оболочке NXT будет отображаться только левый верхний угол передней панели;
- следует убедиться в том, что все нужные регуляторы установлены. И тогда внешний вид передней панели будет выглядеть примерно так:



На четвертом этапе в ***Config Light.vi*** добавляется возможность чтения информации о состоянии регулятора включения датчика:

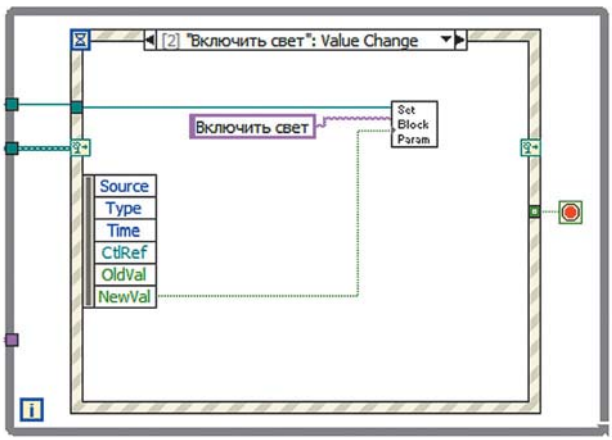
- на поле блок диаграммы находим структуру под названием ***Get Default Values*** from ***Block***;
- удаляем связи между двумя элементами ***Property Node*** и ***Invoke Node***, которые находятся внутри структуры, и вставляем между ними еще один ***Invoke Node***;
- соединяем соответствующие входы и выходы элементов и для нового ***Invoke Node*** выбираем метод ***Get Control Value [Variant]***;
- создаем текстовую константу с текстом ***Включить свет*** и соединяем ее со входом ***Control Name*** нового ***Invoke Node***;
- внутрь структуры помещаем также функцию ***Variant to Data***, и соединяем булевскую константу со входом типа данных;
- создаем локальную переменную для регулятора Включить свет и соединяем ее с выходом ***Variant to Data***.

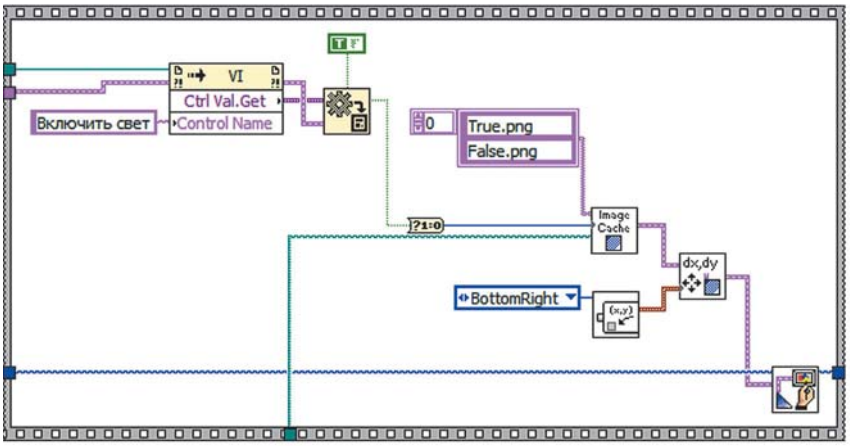
После всех преобразований получим примерно такую структуру:



Шаг пятый. Вносим изменения на поле блок-диаграммы ***Config Light.vi*** в ***Event-структуру***, которая находится справа от только что рассмотренного блока:

- выбираем закладку ***Plug Selector***, нажимаем правой кнопкой мыши и выбираем из меню ***Duplicate Event Case***. В качестве события выбираем изменение значения переменной ***Включить свет***;
- в новом окне ***Event-структуры*** изменяем значение текстовой константы ***Включить свет***. Этот блок программы приобретет следующий вид:





- сохраняем и закрываем ***Config Light.vi***.

Следующий шаг, уже шестой по счету. Добавим изображение дополнительного регулятора путем внесения изменений в программу ***Draw Light.vi***. Для этого Вам понадобятся SubVI, которые находятся в директории ***LabVIEW ≫ vi.lib ≫ addons ≫ NXTToolkit ≫ Block Templates≫ Support***. Итак:

- на поле блок-диаграммы находим пустую ***Sequence***-структуру. Именно в нее нужно добавлять новые элементы;
- помещаем в структуру ***Invoke Node*** с методом ***Get Control Value [Variant]*** и ***Variant to Data***. Как и ранее, создаем текстовую и булевскую константы и соединяем их с новыми элементами;
- из директории, указанной выше, выбираем ***NXT_ImageDataCache.vi*** и помещаем его в структуру. Создаем константу для входа ***Image Name Array*** и указываем ***True.png*** как первый элемент и ***False.png*** как второй. Этот VI хранит изображения в памяти для их мгновенного отображения при необходимости;
- из той же директории "вытягиваем" ***NXT_Block ParamOffset.vi.*** Этот VI определяет местоположение разработанного Вами интерфейса ***(в Config Light.vi)*** при работе в NXT-среде. Создаем константу для входа ***Pad Selector*** и устанавливаем значение ***Bottom Right***;
- помещаем в структуру также функции под названием ***OffsetImageCluster.vi*** и ***Draw Flattened Blended Pixmap.vi***. Они отвечают за правильное расположение графических элементов при работе в NXT-среде. После всех описанных преобразований структура должна иметь вид, как показано выше на диаграмме;
- сохраняем и закрываем файл.

Шаг седьмой и, последний. Импортируем созданный элемент в LEGO® Mindstorms® NXT Software. Перед добавлением новых элементов, необходимо установить утилитку ***Dynamic Block Update*** (динамическое обновление блоков). У Вас ее, конечно же, нет. Но этот софт доступен и находится по адресу - *mindstorms.lego.com/support/updates*. А далее:

- запускаем среду разработки приложений для контроллеров NXT и выбираем ***Tools ≫ Block Import and Export Wizard***;
- нажимаем ***Browse*** и находим директорию, в которой находится импортируемый блок;
- находим этот блок и указываем, в какой группе он будет находиться, после чего нажимаем Import;
- далее нужно перезапустить NXT-софт. Пять минут ожидания и, если все было сделано правильно - Вы увидите созданный блок.

Если Ваш блок недоступен для импорта, нужно проверить все измененные файлы, наверняка где-то закралась ошибка: либо соединительные линии разорваны, либо константы не те, либо еще что-то. Кстати, в первый раз так оно обязательно и будет. Не все так просто, как кажется. Но не отчаивайтесь, ведь у Вас впереди Новогодние каникулы. Времени вполне хватит и на то, чтобы опробовать функции генерации сигналов, и на то, чтобы освоить механизм создания имени себя функциональных блоков для NXT .

Кстати, с Новым Годом и наилучшими пожеланиями!

Материал подготовлен продвинутыми пользователями LabVIEW – студентами старших курсов украинских ВУЗов