Язык программирования EV3 Basic.
Краткий перевод официального руководства

Оригинал: https://sites.google.com/site/ev3basic/ev3-basic-programming

Перевод: Андрей Степанов
«Карандаш и Самоделкин»

I. Экран, кнопки на блоке, подсветка блока, динамик

1. Экран

EV3 имеет черно-белый экран с разрешением 178 x 128 пикселей. Левый верхний угол экран имеет координаты (0,0), правый нижний (178,128).

Для работы с экраном EV3 Бейсик имеет следующие команды:

LCD.BmpFile(цвет, x, y, имя_файла) выводит к заданных координатах картинку из файла RGF. Может содержать путь к файлу. Файл можно создать в официальном ПО от LEGO.
LCD.Clear() Очищает экран
LCD.Circle(цвет, x, y, радиус) Рисует круг заданного радиуса к указанных координатах
LCD.FillCircle(цвет, x, y, радиус) Рисует закрашенный круг заданного радиуса к указанных координатах
LCD.Rect (цвет, x, y, ширина, высота) Рисует прямоугольник заданных размеров к указанных координатах
LCD.FillRect (цвет, x, y, ширина, высота) Рисует закрашенный прямоугольник заданных размеров к указанных координатах
LCD.InverseRect (x, y, ширина, высота) Инвертирует (меняет на противоположный) цвет пикселей в указанной прямоугольной области
LCD.Line (цвет, x1, y1, x2, y2) Рисует линию из точки x1, y1 в точку x2, y2
LCD.Pixel (цвет, x, y) Закрашивает пиксель указанным цветом
LCD.Text (цвет, x, y, размер, "текст") Пишет текст заданного размера и цвета в указанной позиции
LCD.Write (x, y, "текст") Пишет текст среднего размера в позиции x, y
LCD.StopUpdate () Перестает обновлять экран до вызова **LCD.Update,** накапливая изменения
LCD.Update () Выводит все накопленные изменения после **LCD.StopUpdate** на экран

Программирование традиционно начинают с вывода Hello, World!

```
LCD.Clear() ' очищаем экран
LCD.write(45,60,"Hello World")  ' печатаем начиная с точки (45,60) на экране
Program.Delay(10000)    ' ждем 10 секунд прежде чем завершить программу
```

Команда LCD.write печатает текст всегда среднего размера, чтобы напечатать текст меньшего 0, среднего 1 или большого 2 размера изменив в программе оператор на LCD.text

```
LCD.Clear()
LCD.text(1,0,55,2,"Hello World")      '1=черный цвет, пишем начиная с (0,55),  2=размер
Program.Delay(10000)       'ждем чтобы успеть рассмотреть результат
```

Размер 1 и 0 одинаков, но 1 – более жирный. Размер 0 очень мелкий, трудно рассмотреть текст, зато можно уместить 22 символа в строке и около 10 строк на экране при размере

символа 12 пикселей по высоте. Большой шрифт имеет символы в высоту около 20 пикселей, в строке умещается 11 символов и строк может быть до 6. Если Вам нужно напечатать несколько строк текста, хорошая идея выключить обновление экрана функцией **LCD.StopUpdate**, напечатать текст, затем включить обновление функцией LCD**.Update** выведя весь текст за раз.

2. Кнопки на блоке

Блок EV3 имеет 6 кнопок, но только 5 из них может использовать EV3 Бейсик. Шестая кнопка (Назад) завершает выполняющуюся программу. Кнопки обозначаются так: Вверх (U), Вниз (D), Влево (L), Вправо (R) и Ввод (E). Для работы с кнопками имеются следующие команды:

Buttons.Current Возвращает коды кнопок в моменты, когда они нажаты. Например, если вызвать функция в момент, когда Вы удерживаете Влево и Вверх, то функция выдаст строку "LU".
Buttons.Wait() waits Ждет, пока одна из кнопок не будет нажата и отпущена.
Buttons.GetClicks() Проверяет, какие кнопки были нажали с момента последнего вызова GetClicks и возвращает строку, с их перечислением
Buttons.Flush() Удаляет из памяти состояния всех кнопок. Последующие вызовы GetClicks выдаст только кнопки, которые были нажаты после вызова этой функции.

Следующая программа ждет нажатия кнопок и выводит на экран соответствующие им символы

```
Buttons.Wait()
LCD.Clear()
LCD.Text(1,70,35,2,"You")
LCD.Text(1,35,55,2,"pressed")
LCD.Text(1,80,85,2,Buttons.GetClicks())
Program.Delay(6000)
```

3. Подсветка на блоке

EV3 блок имеет подсветку, расположенную под кнопками. Она может загораться зеленым, красным и оранжевым цветом. Есть всего лишь одна команда для управления подсветкой на блоке

EV3.SetLEDColor (Цвет, Эффект) Управляет подсветкой и ее режимом
Цвет может быть "OFF" (выключена), "GREEN" (зеленый), "RED" (красный) or "ORANGE" (оранжевый).
Эффект может быть "NORMAL" (горит постоянно), "FLASH" (вспыхивает) or "PULSE" (пульсирует).

4. Динамик

В EV3 Бейсик есть ряд команд для работы с динамиком. По умолчанию при вызове команды, проигрывающей звук программа продолжается и не ждем, пока проигрывание закончится. Чтобы дождаться проигрывания воспользуйтесь командой: **Speaker.Wait().**

Speaker.Tone(громкость, частота, длительность) Проигрывает тон указанной частоты. Громкость 0-100, частота в герцах 250-10000. Длительность в миллисекундах.
Speaker.Note(громкость, нота, длительность) Тоже самое, что и предыдущая, но воспроизводит ноту, от ДО 4 октавы "C4" to СИ седьмой "B7" и полутона в форме "C#5".
Speaker.Play(громкость, имя_файла) Проигрывает звуковой файл формата RSF, имеющийся на блоке. Может содержать путь к файлу. Файл можно создать в официальном ПО от LEGO.
Speaker.Stop() останавливает проигрывающийся звук, файл или тон.
Speaker.Wait() Ждет, пока текущий проигрывающийся звук не закончится.

Пример программы по главам 1-4. Вы можете скопировать ее в EV3 Basic и запустить, подключив блок EV3. Чтобы программа автоматически красиво отформатировалась Щелкните на код правой кнопкой мыши и выберите «Format Program» .

****

```
LCD.Clear()
LCD.Write(15,55, "Press buttons other")
LCD.Write(40,70, "than 'Back'")

While "True"
 Program.Delay(200)
 'Проверяем пять раз в секунду какие кнопки нажаты

  click = Buttons.GetClicks()   'переменная 'click' содержит символы
'кнопок, нажатых и отпущенных с момент последнего вызова этой функции
  If click <> "" Then  'Очищаем экран если кнопка нажата
    LCD.Clear()
  endif

  If Text.IsSubText(click, "U") then  'Если в строке есть символ "U" (Вверх)
    LCD.Write(0,0, "This is written at")
    LCD.Write(0,10, "(0,0) using LED.Write")
    EV3.SetLEDColor ("OFF", "Normal")
    Speaker.Tone(100, 500, 400)  'Играем 500 Гц тон 0.4 секунды на полной громкости

  ElseIf Text.IsSubText(click,"D") then
    LCD.Text(1, 0,12, 0, "This is written at")
    LCD.Text(1, 0,24, 0, "(0,12) using LED.Text")
    LCD.Text(1, 0,36, 0, "in size 0 (small).")
    EV3.SetLEDColor ("RED", "Flash")
    Speaker.Note(50, "D2", 400)  'Играем ноту"D2" 0.4 секунды на 50% громкости

  ElseIf Text.IsSubText(click,"L") then
    LCD.Text(1, 0,12, 1, "This is written at")
    LCD.Text(1, 0,24, 1, "(0,12) using LED.Text")
    LCD.Text(1, 0,36, 1, "in size 1 (medium).")
    EV3.SetLEDColor ("GREEN", "Pulse") 'Включаем зеленую пульсирующую подсветку

  ElseIf Text.IsSubText(click,"R") then
    LCD.Text(1, 0,20, 2, "This is")
    LCD.Text(1, 0,40, 2, "made using")
    LCD.Text(1, 0,60, 2, "LED.Text in")
```

```
   LCD.Text(1, 0,80, 2, "size 2")
   LCD.Text(1, 0,100, 2, "(large).")
   EV3.SetLEDColor ("ORANGE", "Normal")  'Включаем красную подсветку

  ElseIf Text.IsSubText(click,"E") then
   LCD.Write(0,0,"Circle(1,50,70,30)")
   LCD.Circle(1, 50, 70, 30)  ' color,x,y,radius
   LCD.Write(0,12,"FillRect(1,90,80,60,40)")
   LCD.FillRect (1, 90, 80, 60, 40)   ' color,x,y,width,height
   LCD.Write(0,24,"Line(1,100,50,160,60)")
   LCD.Line(1,100,50,160,60)   ' color,x1,y1,x2,y2
   for i=1 to 3
    EV3.SetLEDColor ("RED", "Normal")
    Program.Delay(500)  'wait 0.5 seconds (500 milliseconds)
    EV3.SetLEDColor ("Green", "Normal")
    Program.Delay(500)
    EV3.SetLEDColor ("Orange", "Normal")
    Program.Delay(500)
   endfor
  endif
EndWhile

'************************************************************
```

5. Пользовательские картинки и звуки

EV3 Бейсик может работать только с LEGO-форматами звуков RSF и графики RGF/ Звуки и графика должны быть помещены в папку с проектом с помощью EV3 Explorer. Можно использовать звуки и графику из других проектов, указав в имени файла путь, включающий имя проекта, из которого заимствуются ресурсы. Файл можно создать в официальном ПО от LEGO, нарисовав, записав на микрофон, скачав из интернета. Пример использования пути к файлу:

```
LCD.Clear()
LCD.BmpFile(1, 0, 0, "Images/Angry")  'цвет 1 - черный, x=0,y=0, файл Angry из проекта (папки) Images
Speaker.Play(100, "Sounds/T-rex roar") ' расширение файла не указывается
Program.Delay(10000) ' ждем 10 секунд
```

II. Использование моторов

EV3 Бейсик совместим со средними и большими моторами EV3, а также с моторами NXT и, по большому счету, не делает различий при работе с ними.

EV3 Бейсик имеет 9 команд, которые могут использоваться для управления моторами и только 4 из них достаточно простые, чтобы использоваться начинающими робототехниками: Motor.Move, Motor.MoveSync, Motor.Start и Motor.StartSync. И, конечно же, Вам потребуется Motor.Stop для того, чтобы останавливать мотор.

Команды для работы с моторами используют следующие параметры:

порт – порт EV3, к которому подключен мотор, например "BC". "A". Если моторов в параметре несколько – они всегда в алфавитном порядке.
угол – угол поворота мотора. Всегда положительное значение, в случае отрицательного – знак игнорируется. Если мотор нужно вращать в обратную сторону – меняйте знак у скорости, а не у угла! Конвертация оборотов в градусы – умножением на 360 и наоборот.
тормоз = "True", когда после остановки мотор должен затормозить, иначе "False"
скорость – от -100 до 100, знак числа определяет направление

Примеры использования при повороте мотора на определенный угол:

Motor.Move(порт, скорость, угол, тормоз) - для поворота одного мотор на определенный угол
Пример: **Motor.Move("A", 40, 80, "True")** – мотор А на скорости 40 повернуть на 80 градусов и затормозить.

Motor.Move(порты, скорость, угол, тормоз) - для поворота нескольких мотор на определенный угол с одинаковой скоростью
Пример: **Motor.Move("BC", -30, 720, "True")** - моторы B и C на скорости -30 повернуть на 2 оборота (720 градусов) и затормозить.

Motor.MoveSync(порты, скорость1, скорость2, угол, тормоз) – команда используется для того, чтобы робот мог двигаться по изогнутой линии или разворачиваться на месте используя два мотора с разными скоростями вращения, контролируя угол поворота мотора. Скорость1 и скорсоть2 – скорости моторов, от меньшего по алфавиту порта к большему. Угол – угол, на который должен повернуться более быстрый из двух моторов. Команда похожа на блок «Танк» в стандартном ПО LEGO EV3-G/
Пример: **Motor.MoveSync("BC", 30, 40, 720, "True")** – моторы B и C включить со скоростями 30 и 40 до того момента, пока мотор C не сделает два оборота (720 градусов), затем затормозить.

Примеры использования при работе мотора заданный промежуток времени:

Чтобы мотор работал определенный промежуток времени нужно запустить его, подождать командой Program.Delay и выключить командой Motor.Stop

Motor.Start(порты, скорость) - чтобы включить один или более моторов определенной (одинаковой для нескольких) скоростью.
Пример: включаем моторы B и C на 3 секунды со скоростью 80

```
Motor.Start("BC",80)
Program.Delay(3000)   ' ждем 3 секунды
Motor.Stop("BC","True")    ' "True" для торможения
```

Motor.StartSync(порты, скорость1, скорость2) - команда используется для того, чтобы робот мог двигаться по изогнутой линии или разворачиваться на месте используя два мотора с разными скоростями вращения.
Пример: робот разворачивается на месте 5 секунд, затем останавливается

```
Motor.StartSync("BC", 50, -50)
Program.Delay(5000)
Motor.Stop("BC","True")
```

Изменение скорости: если мотор уже вращается с какой-то скоростью, то команда Motor.Start изменит его скорость на новую, заданную в ней.

В следующей таблице приведены рекомендации по выбору команд для управления моторами

	Двигаться x градусов, программа ждет завершения	**Включиться на постоянное вращение**	**Запуститься для вращения x градусов**
Регулировка скорости	Motor.Move	Motor.Start	Motor.Schedule
Регулировка мощности	Motor.MovePower	Motor.StartPower	Motor.SchedulePower
Синхронизация моторов	Motor.MoveSync	Motor.StartSync	Motor.ScheduleSync

Другие команды:

Motor.GetCount (порт) – прочитать показания энкодера на порту
Motor.ResetCount (порты) – сбросить показания энкодера на портах
Motor.GetSpeed (порт) – выдает скорость мотора, подключенного к порту
Motor.IsBusy (порты) – выдает True", если один или несколкьомоторов заняты, иначе "False"
Motor.MovePower (порты, мощность, угол, торможение) – тоже самое, что и **Motor.Move**, но вместо скорсоти мотора оперирует его мощностью
Motor.Schedule (порты, скорость, угол1, угол2, угол3, тормоз) – поворот на заданный угол с плавным стартом и остановом. Общий угол поворота – угол1+угол2+угол3, причем на участке угол1 происходит ускорение, на участке угол2 – заданная скорость поддерживается, на участке угол3 – происходит замедление. Важно! Команда не ждет завершения работы мотора. Если енобходимо дождаться выполнения, используйте Motor.Wait(порты)
Motor.SchedulePower(порты, мощность, угол1, угол2, угол3, тормоз) – тоже самое, что и предыдущая, но оперирует мощностью вместо скорости.
Motor.Wait(порты) – ожидает, пока **Schedule** или **Move** команда не закончит свою работу

Пример программ для управления моторами:

```
'**********************************************************

' Простое управление моторами, подключенными к портам B и C
' У вас может оказаться реверсивное подключение моторов, поэтому
' если наблюдаете движение робота в обратном направлении
' измените знак у параметра скорости

' Робот поворачивает направо используя мотор B
' включенный на 50% скорости на угол 360°, затем тормозит
Motor.Move ("B", 50, 360, "True")
Program.Delay(2000)  ' Пауза 2 секунды (2000 миллисекунд)

' Робот движется вперед на 30% скорости на два оборота мотора
```

' (720°), затем тормозит
Motor.Move ("BC", 30, 720, "True")
Program.Delay(2000)

' Чтобы выполнить старт и останов более плавным используем команды 'Schedule' или
' SchedulePower. Эти команды имеют зону (угол поворота мотора) разгона,
' зону постоянной скорости и зону торможения. Для примера
' Motor.Schedule (порты, скорость, зона1, зона2, зона3, торможение)
' зона1 and зона3 это углы, в пределах которых мотор загоняется и замедляется

' Чтоыб разогнать робота в течении одного оборота моторов,
' затем поддерживать скорость 60% в течении двух оборотов моторов (720°),
' затем замедляться в течении трех оборотов(1080°) при движении вперед:
Motor.Schedule("BC", 60, 360, 720, 1080, "True")
Motor.Wait ("BC")
' Замечание – команда Schedule не ждет пока мотор отработает заданное действие,
' поэтому если Вам не нужно в это время выполнять других действий воспользуйтесь
' командой Motor.Wait() чтобы дождаться окончания команды Schedule
Program.Delay(2000)

' Если вы применяете команду Motor.Move() к нескольким моторам
' они будут вращаться с одинаковой скоростью. Как же можно
' заставить их вращаться с разными скоростями?

' В стандартном графическом ПО от LEGO есть программный блок “Move Tank”
' для вращения двух моторов с разными скоростями на определенное количество
' градусов, оборотов или по времени.

' В среде EV3 Basic заменой такому танковому блоку, управляющему
' двумя моторами для движения по искривленной траектории
' с разными скоростями на каждом из моторов является команда
' Motor.MoveSync(порты, скорость1, скорость2, угол, тормоз)
' Скорость1 и скорость2 – скорости двух моторов, перечисленных в портах
' в алфавитном порядке, угол – для более быстрого из двух моторов

Motor.MoveSync("BC", 50, -60, 215, "True")
' В этом примере мотор B повернется со скоростью 50%, а мотор C со скоростью -60%
' в обратном направлении на 215 градусов, считая по мотору C, т.к. он быстрее
' Для разворота робота на месте скорости моторов должны быть равны, но
' противоположны по знаку.
' Motor.MoveSync заставляет программу ждать, пока она выполняется

Program.Delay(2000)

' Ниже приведен пример программы для рисования роботом квадрата
' Возможно Вам придется изменить угол 430° в зависимости от конструкции робота
' Чтобы заставить повернуть его на 90°

For i=1 to 4
** Motor.Move ("BC", 40, 720, "True")**
** Motor.Move ("B", 40, 430, "True")**
EndFor

```
Program.Delay(2000)

' Приведенный выше пример ведет робота по квадрату с закругленными углами
' так как робот не разворачивается на месте, а вращает одним мотором
' В следующем примере робот проедет по квадрату с поворотами в его углах
' вокруг своей оси, для этого его моторы будут вращаться в противоположных
' направлениях

For i=1 to 4
  Motor.Move ("BC", 40, 720, "True")   'Move forwards
  Motor.MoveSync("BC", 40, -40, 215, "True")
EndFor

'******************************************************
' В следующем роботе робот едет по правильному многоугольнику,
' Число сторон которого вводит пользователь в появившемся текстовом окне

Textwindow.Write ("Trace a polygon with how many sides (3-9)?")
sides=textwindow.ReadNumber()
degrees=2.39*360/sides 'Каждое колесо должно повернуться на этот угол
' Поскольку при повороте каждого колеса на 215° робот поворачивал на 90°
' мы должны повернуть каждое колесо на 215/90=2.39° чтобы робот повернул на 1°.

For i=1 to sides
  Motor.Move("BC", 50, 720, "True")
  Program.Delay(500)
  Motor.MoveSync("BC", 20, -20, degrees, "True")
  Program.Delay(500)
EndFor

'******************************************************
' Следующая программа использует кнопки EV3 для управления
' моторами, подключенными к портам B и C.

LCD.Clear()
LCD.Text(1, 0,30, 1, "Control motors with")
LCD.Text(1, 0,45, 1, "directional buttons")
LCD.Text(1, 0,70, 1, "Left/Right: Motor B")
LCD.Text(1, 0,85, 1, "Up/Down: Motor C")

while "True"  '"True" это всегда истина, этот цикл будет бесконечен
  K = Buttons.Current  ' запросим состояние кнопок

  SB = 0   ' установим скорость мотора B в 0
  SC = 0   ' установим скорость мотора C в 0
  If Text.IsSubText(K, "L") then
' проверяем нет ли в строке символа L, то есть не нажата ли кнопка Влево
    SB = -40
  elseif Text.IsSubText(K, "R") then
    SB = 40
  endif
  If Text.IsSubText(K, "U") then
```

```
    SC = 40
  elseif Text.IsSubText(K, "D") then
    SC = -40
  endif

  ' Устанавливаем скорости моторам
  Motor.Start("B", SB)
  Motor.Start("C", SC)

  ' ждем 100 миллисекунд, чтобы не сильно нагружать опросом кнопок блок
  Program.Delay(100)
Endwhile
```

III. Датчики

EV3 Бейсик совместим со всеми стандартными EV3 и NXT датчиками, правда датчик звука пока под вопросом, работа по его поддержке продолжается.

Как и в LEGO EV3-G многие датчики могут использоваться в нескольких режимах. Чтобы переключить датчик в определенный режим используется команда **Sensor.Setmode(порт, режим)**, например команда **Sensor.SetMode(3, 1)** переключит датчик на порту 3 в режим Важно всегда устанавливать режим работы датчика перед его использованием..

- Некоторые режимы датчиков возвращают значение в процентах. Для чтения показаний таких датчиков используйте функцию **Sensor.ReadPercent(порт).** К примеру датчик касания возвращает в этом режиме 0, если кнопка не нажата и 100 в нажатом положении.
- Некоторые режимы датчиков возвращают значение не в процентах. Например, датчик цвета в режиме 2 «определение цвета» возвращает коды цветов от 0 до 7 (0 – неизвестно, 1 – черный и т.д.), ультразвуковой датчик в режиме 0 выдает целые числа от 0 до 255, соответствующие измеренному им расстоянию до объекта в миллиметрах. Для чтения одного значения, не в процентах, используйте функцию **Sensor.ReadRawValue(порт, индекс).** Индекс как правило равен 0, за исключением особых случаях, о которых – ниже.
- Некоторые режиме датчиков возвращают несколько значений одновременно, в массиве. Чтобы прочитать этот массив используйте функцию **Sensor.ReadRaw(порт, количество_элементов).** Например, чтобы получить RGB (красную, зеленую и синюю составляющие цвета) с датчика цвета на порту 3 используйте функцию **Sensor.ReadRaw(3, 3).** Нумерация элементов в массиве начинается с нуля. Первый элемент имеет индекс [0], это интенсивность красной компоненты света, [1] – зеленой и [2] – синей.

1. Датчик касания (кнопка)

Датчик касания (кнопка) используется с функцией **Sensor.ReadPercent(порт), которая** возвращает 0, если кнопка не нажата и 100 в нажатом положении.

2. Цветосветовой датчик

В EV3 Бейсике цветосветовой датчик может работать в 4 режимах:

- Режим отраженного света (режим 0)
- Режим измерения уровня внешней освещенности (режим 1)
- Режим измерения цвета (режим 2)
- Режим измерения RBG-составляющих цвета (режим 4)

В режиме 0 датчик возвращает по функции ReadPercent**(порт)** уровень отраженного света в процентах (от 0 до 100).

В режиме 1 датчик возвращает 0 при минимуме внешнего освещения и 199 на ярком свету, используется функция **ReadPercent(порт).** Ниже пример программы, выводящей на экран уровень внешней освещенности в %

```
Sensor.SetMode(3,1)  ' переводим датчик цвета на 3 порту в режим 1
While "True"
   LCD.StopUpdate()    ' не обновляем экран, пока не подготовим вывод текста
   LCD.Clear()
   LCD.Text(1,30,20, 2, "Ambient")
   LCD.Text(1,40,40, 2, "light")
   LCD.Text(1,10,60, 2, "intensity:")
   LCD.Text(1,80,90, 2, Sensor.ReadPercent(3))
   LCD.Update()
   Program.Delay(100)  ' ждем 0.1 секунду
EndWhile
```

В режиме 2 датчик цвета возвращает код цвета (0 – цвет не определен, 1 – черный, 2 – синий, 3 – зеленый, 4 – желтый, 5 – красный, 6 – белый, 7 – коричневый). Для работы с датчиком в этом режиме используйте функцию Sensor.ReadRawValue(порт, 0). Ниже приведен пример программы, отображающей на экране название цвета, который определил датчик.

```
Sensor.SetMode(3,2)  'Устанавливаем режим 2 датчика цвета на порту 3

While "True"
   LCD.StopUpdate()    'не обновлять экран пока не выведем на него текст
   LCD.Clear()
   code=Sensor.ReadRawValue(3, 0)
   LCD.Text(1,33,40, 2, "Color "+ code)
   If code =0 Then
      col="UNKNOWN"
   ElseIf code =1 Then
      col="BLACK"
   ElseIf code =2 Then
      col="BLUE"
   ElseIf code =3 Then
      col="GREEN"
   ElseIf code =4 Then
      col="YELLOW"
   ElseIf code =5 Then
      col="RED"
   ElseIf code =6 Then
      col="WHITE"
   ElseIf code =7 then
```

```
      col="BROWN"
   EndIf
   LCD.Text(1,33,75, 2, col)
   LCD.Update()
   Program.Delay(100)
EndWhile
```

Эту же самую программу можно написать красивее, используя массив цветов.

```
Sensor.SetMode(3,2)

' Заполняем массив цветов
Colors[0]="UNKNOWN"
Colors[1]="BLACK"
Colors[2]="BLUE"
Colors[3]="GREEN"
Colors[4]="YELLOW"
Colors[5]="RED"
Colors[6]="WHITE"
Colors[7]="BROWN"

While "True"
   LCD.StopUpdate()
   LCD.Clear()
   code=Sensor.ReadRawValue(3, 0)
   LCD.Text(1,33,40, 2, "Color "+ Code)
   LCD.Text(1,33,75, 2, Colors[Code])
   LCD.Update()
   Program.Delay(100)
EndWhile
```

Обратите внимание, датчик цвета LEGO откалиброван на цвета кубиков LEGO, по остальным оттенкам цветов, даже если они кажутся для вас очевидными, датчик может выдать неожиданный результат.

Программа с массивом, указанная выше будет прекрасно работать в режиме, когда вы запустите ее с ПК из Small Basic, но не запустится автономно на блоке EV3. Это связано с ограничениями прошивки LEGO, в данном случае связанными с невозможностью работать с массивами строк.

В режиме 4 (не 3! третьего режима у датчика цвета нет) датчик цвета возвращает массив из RBG-составляющих цвета. Это позволит Вам определить любой оттенок цвета, ориентируясь на его составляющие.

3. Ультразвуковой датчик

Для определения расстояния, которое возвращает ультразвуковой датчик используйте функцию **Sensor.ReadRawValue(port number, 0)**. Датчик ультразвука EV3 в режиме 0 возвращает расстояние **в миллиметрах** (не в сантиметрах!). Датчик ультразвука NXT в режиме 0 возвращает расстояние **в сантиметрах.** Датчик EV3 в режиме 1 возвращает расстояние в десятых дюймах.

Ниже приведена программа, которая постоянно отображает на экране расстояние, измеренное ультразвуковым датчиком EV3, переводя его в сантиметры делением на 10:

```
' Ультразвуковой датчик подключен к 4 порту
Sensor.SetMode(4,0) ' устанавливаем порт 4 в режим 0 – измерения в расстоянии в мм

While "True"
   LCD.StopUpdate() ' для избежание мерцания не обновляем экран пока текст не готов
   LCD.Clear()
   LCD.Text(1,45,55,2,Sensor.ReadRawValue(4, 0)/10+" cm")  ' Переводим мм в см
   LCD.Update()
EndWhile
```

4. Инфракрасный датчик

В EV3 Бейсике инфракрасный датчик из домашней версии набора EV3 может работать в следующих режимах:

Режим 0 – измерение расстояния до объекта в см
Режим 1 – измерение расстояния и направления на ИК-маяк
Режим 2 – сигналы, принятые от ИК-маяка (или от маяков, до 4 одновременно)

По умолчанию датчик работает в режиме 0 и функция Sensor.ReadPercent() возвращает целое число от 0 до 100 - расстояния до объекта в см. Это расстояние не особо точное, зависит от освещенности объекта. Более точно измеряется расстояние до ярко освещенных объектов.

Ниже приведена программа, которая постоянно отображает на экране расстояние, измеренное инфракрасным датчиком в сантиметрах:

```
' Подключите ИК-датчик к 4 порту
Sensor.SetMode(4,0)  ' устанавливаем режим работы 4 порта в 0, для измерения в см

While "True"
   LCD.StopUpdate() ' для избежание мерцания не обновляем экран пока текст не готов
   LCD.Clear()
   LCD.Text(1,45,55,2,Sensor.ReadPercent(4)+" cm")
   LCD.Update()
EndWhile
```

При переключении в режим 1 с помощью команды **Sensor.SetMode(4,1)** ИК датчик на 4 порту начинает возвращать расстояние и направление на ИК-маяк. При этом он возвращает оба значения одновременно по функции **Sensor.ReadRaw(порт, 2)**. Возвращаемое ей значение – массив, в 0 элементе – направление на ИК маяк, в 1 элементе - расстояние до ИК-маяка в см.

После переключения в режим 2 **IR-REMOTE**, с помощью команды.SetMode(4,2) ИК-датчик, подключенный в порт 4, начинает определять, какие кнопки нажаты на удаленном маяке (маяках).

В случае, если только один маяк передает коды нажатых кнопок на ИК-датчик, необходимо использовать **Sensor.ReadRawValue(4, канал_передачи)** для получения кодов кнопок.

A - левая верхняя, B – левая нижняя, C – правая верхняя, D – правая нижняя. E - средняя
Коды, принимаемые **Sensor.ReadRawValue(4, канал_передачи)**

A = 1	A и B = 10	B и C = 7
B = 2	A и C = 5	B и D = 8
C = 3	A и D = 6	C и D = 11
D = 4	E = 9	

Все другие комбинации выдадут 0

В случае, если несколько маяков одновременно передают коды нажатых кнопок на ИК-датчик на разных каналах, необходимо использовать функцию **Sensor.ReadRaw(4, 4)**, которая возвращает массив из 4 значений, одного на каждый канал. 0-й элемент массива соответствует маяку на 1 канале и т.д.

5. Гироскоп

Датчик гироскопа входит в состав образовательного набора EV3, он может быть приобретен отдельно для домашней версии, с которой полностью совместим.

Очень важно, чтобы датчик гироскопа был неподвижен в момент запуска блока EV3 или при подключении кабеля между гироскопом и включенным блоком, иначе показания гироскопа будут постоянно отклоняться (явление «дрейфа»). Для устранения дрейфа можно переподключить кабель к неподвижному датчику. Проверить, проявляется ли дрейф можно в меню «Port View»? показания дна порту с гироскопом должны быть стабильны при неподвижном датчике.

Режим 0 гироскопа измеряет угов в градусах относительно позиции датчика на момент старта программы или сброса его показаний. Датчик одноосный, может измерять отклонения или в горизонтальной или в вертикальной плоскости – зависит от того как Вы установите его на робота. Для чтения показаний в этом режиме используйте функцию **Sensor.ReadRawValue(порт, 0)** которая возвращает массив из единственного 0-го элемента.

Ниже приведена программа, отображающая на экране показания гироскопа в градусах

```
' Гироскоп подключен к 2 порту
Sensor.SetMode(2,0)  ' Режим 0 – измерение отклонения в градусах

While "True"
  LCD.StopUpdate()   ' для избежание мерцания не обновляем экран пока текст не готов
  LCD.Clear()
  LCD.Text(1,35,50,2, Sensor.ReadRawValue(2,0) + " deg")
  LCD.Update()
  Program.Delay(100)  ' ждем 0.1 секунду
EndWhile
```

Режим 1 гироскопа измеряет скорость изменения отклонения в градусах в секунду. Предыдущая программа подходит для примера работы в этом режиме, замените в ней выбор режима работы на **Sensor.SetMode(2,1)** и увидите на экране скорость изменения отклонения в градусах в секунду.

Обратите внимание, что несмотря на то, что датчик гироскопа измеряет отклонения в градусах, встроенные тригонометрические функции Small Basic используют радианы, радиан - это примерно 57.3°.

6. Мотор как датчик угла

Также как и в официальном ПО от LEGO моторы могут использоваться в качестве датчиков угла. Функция **Motor.GetCount (порт)** показывает в градусах угол поворота оси мотора, подключенного к порту, указанному в ее параметрах.

7. Перечень режимов датчиков

EV3 датчики

Тип	Режим	Название режима	Доступные функции	Возвращаемое значение
16	0	TOUCH	ReadPercent	0=не нажат, 100=нажат
29	0	COL-REFLECT	ReadPercent	0=нет отраженного света, 100=макс
29	1	COL-AMBIENT	ReadPercent	0=нет внеш. освещ., 100=макс
29	2	COL-COLOR	ReadRawValue	0=неизвестно, 1=черный, 2=синий, 3=зеленый, 4=желный, 5=красный, 6=белый, 7=коричневый
29	4	RGB-RAW	ReadRaw (3 vals)	value0=красная составляющая, value1=зеленая составляющая, value2=синяя составляющая
30	0	US-DIST-CM	ReadRawValue	Расстояние в миллиметрах
30	1	US-DIST-IN	ReadRawValue	Расстояние в десятых дюйма
32	0	GYRO-ANG	ReadRawValue	Угол отклонения в градусах
32	1	GYRO-RATE	ReadRawValue	Скорость изменения угла отклонения в градусах в секунду
33	0	IR-PROX	ReadPercent	Расстояние в см (примерное)
33	1	IR-SEEK	ReadRaw (2 vals)	value0=направление на ИК-маяк value1=Расстояние до ИК-маяка
33	2	IR-REMOTE	ReadRawValue***	value0=сигнал на канал 1 value1= сигнал на канал 2…

NXT датчики

Тип	Режим	Название режима	Доступные функции	Возвращаемое значение
1	0	NXT-TOUCH	ReadPercent	0=не нажат, 100=нажат
4	0	NXT-REFLECT	ReadPercent	0=нет отраженного света, 100=макс
4	1	NXT-AMBIENT	ReadPercent	0=нет внеш. освещ., 100=макс
4	2	NXT-COLOR	ReadRawValue	1=черный, 2=синий, 3=зеленый, 4=желтый, 5=красный, 6=белый
5	0	NXT-US-CM	ReadRawValue	Расстояние в сантиметрах

IV. Другие полезные функции

1. Удаленное управление с компьютера

Данная программа позволяет с клавиатуры компьютера управлять колесным роботом.

```
' подключите моторы робота к портам B и C
Textwindow.Writeline("Нажмите 8 чтобы ехать вперед очень быстро")
Textwindow.Writeline("Нажмите 5 чтобы ехать вперед с нормальной скоростью")
Textwindow.Writeline("Нажмите 4 для поворота налево")
Textwindow.Writeline("Нажмите 6 для поворота направо")
Textwindow.Writeline("Нажмите 2 для движения назад")
Textwindow.Writeline("Нажмите 0 (ноль) для остановки")
Textwindow.Writeline("Нажимайте, а не удерживайте нажатыми кнопки!")
Textwindow.Writeline("Вы можете попробовать еще кнопки 7, 9, 1, и 3")
Textwindow.Writeline("")

' следующие два массива задают скорости моторов B и C
b_speed="0=0;1=-20;2=-30;3=0;4=-10;5=40;6=10;7=0;8=80;9=20"
c_speed="0=0;1=0;2=-30;3=-20;4=10;5=40;6=-10;7=20;8=80;9=0"

While "True"   ' повторять бесконечно
   inputletter=TextWindow.ReadKey()
   Motor.StartSync("BC", b_speed[inputletter], c_speed[inputletter])
   TextWindow.Write (inputletter)
EndWhile
```

Пример программы для управления роботом мышью с компьютера

```
' подключите моторы робота к портам B и C
GraphicsWindow.WIDTH=400
GraphicsWindow.HEIGHT=400
GraphicsWindow.DrawLine(200,0,200,400 )
GraphicsWindow.DrawLine(0,200,400 ,200 )

While "True"    ' повторять бесконечно
   x = (GraphicsWindow.MouseX-200)/2
   y = (200-GraphicsWindow.MouseY)/2
   b_speed = (x+y)/2     ' вычисляем скорость мотора B
   c_speed = (y-x)/2     ' вычисляем скорсоть мотора C
   Motor.StartSync("BC", b_speed, c_speed)
   Program.Delay(100)
EndWhile
```

2. Параллельные задачи (потоки)

Поток представляет собой фрагмент программного кода, который может работать независимо и параллельно основной программе. Например ты можешь создать поток, который будет управлять двигателями в то время как основная программа будет опрашивать датчики или ожидать действий пользователя.

Ниже приведен пример двухпоточной программы BLINKER, в ней робот выполняет две независимые задачи – нигает подсветкой с частотой 1 раз в секунду и выводит текст на экран с частотой 0,4 секунды:

```
LCD.Clear()
Thread.Run = BLINKER

Sub BLINKER
   While "true"
      EV3.SetLEDColor("ORANGE","NORMAL")
      Program.Delay(500)
      EV3.SetLEDColor("OFF","NORMAL")
      Program.Delay(500)
   EndWhile
EndSub

For y=0 to 120 Step 10  ' цикл запустится 13 раз, не 12
   LCD.Text(1,0,y,1,"Tick")
   Program.Delay(400)  'pause 0.4 seconds
EndFor
```

3. Подпрограммы

Подпрограммы логично использовать в больших программах, когда вам часто нужно выполнять одни и те же действия, либо для оформления подпрограммы в качестве параллельного потока. Объявление подпрограммы в общем виде выглядит так:

```
Sub test   ' test – имя подпрограммы
  ' здесь инструкции подпрограммы
EndSub
```

Слово Sub (от англ. subprogram — подпрограмма) показывает, что далее следуют инструкции подпрограммы, слово EndSub отмечает конец подпрограммы. Идентификатор Имя определяет имя подпрограммы.

Вызвать подпрограмму очень просто. Нужно всего лишь указать ее имя следующим образом:

```
test ()
```

Скобки означают, что это не переменная, а именно вызов подпрограммы. Сам код подпрограммы может быть написан в любом месте вашей программы, при вызове подпрограммы это не имеет значения.

4. Ввод текста на блоке EV3

Стандартное ПО LEGO EV3-G имеет крайне скудный набор команд для работы с текстовой информацией, по сути это всего лишь одна команда «Сцепить.»

Следующий пример на EV3 Basic показывает, как на блоке EV3 может происходить автономный ввод текстовой информации. Программа может работать как в «PC-режиме», так в автономном с компиляцией для работы на блоке.


```
mytext=""
LCD.Clear()
LCD.Text(1,0,3,1," <=Backspace  >=Enter")
line[1]="0123456789."
line[2]="ABCDEFGHIJK"
line[3]="LMNOPQRSTUV"
line[4]="WXYZ -  < >"
LCD.Text(1,0,16,2,line[1])
LCD.Text(1,0,38,2,line[2])
LCD.Text(1,0,60,2,line[3])
LCD.Text(1,0,82,2,line[4])
LCD.Rect(1,0,103,178,25)  ' рисуем прямоугольник
col=6   ' текущий столбец
row=2   ' текущая строка
x=81    'x координата выделенного символа 'F'
y=35    'y координата выделенного символа 'F'
LCD.InverseRect(x,y,16,22)   'инвертируем символ 'F'

loop="True"   ' цикл, пока переменная истинна
While loop
  Buttons.Wait()
  Button=Buttons.GetClicks()
  LCD.InverseRect(x,y,16,22)   ' очистить инверсию символа
  If Button="R" Then
    col=1+Math.Remainder(col,11)  ' прыжок из столбца 11 к 1
  ElseIf Button="L" Then
    col=11-Math.Remainder(12-col,11) 'прыжок из столбца 1 к 11
  ElseIf Button="U" Then
    row=4-Math.Remainder(5-row,4)  ' прыжок со строки 4 к 1
```

```
  ElseIf Button="D" Then
    row=1+Math.Remainder(row,4)  ' прыжок со строки 1 к 4
  ElseIf Button="E" Then
    If row=4 and col>6 Then
      If col =9 then   'backspace
        If  mytext<>"" Then
          ' удалить последнйи символ из mytext
          mytext=Text.GetSubText(mytext,1,Text.GetLength(mytext)-1)
        Else  ' нечего удалять
          Speaker.Tone(100,1000,1000)
        endif
      ElseIf col = 11 Then   'Enter
        Loop="False"   ' не повторять цикл
      Else   'col=7,8 или 10 не доступны с строке 4
        Speaker.Tone(100,1000,1000)
      EndIf
    Else  'средняя кнопка нажата
      If Text.GetLength(mytext)<>11 Then
        mytext=mytext+Text.GetSubText(line[row],col,1)
      Else  ' уже введено 11 символов
        Speaker.Tone(100,1000,1000)
      EndIf
    EndIf
  EndIf
  x=-15+col*16 ' рассчитать координату x символа для подсвечивания
  y=-9+row*22  ' рассчитать координату y символа для подсвечивания
  LCD.InverseRect(x,y,16,22)
  LCD.Text(1,1,107,2,"          ")  ' очистить текстовую строку
  LCD.Text(1,1,107,2,mytext)
EndWhile

LCD.Clear()
LCD.Text(1,0,16,2,"Here is")
LCD.Text(1,0,38,2,"your text:")
LCD.Text(1,0,82,2,mytext)
Program.Delay(8000)   ' отображаем в течении 8 секунд
```

4. Ошибки при компиляция программы на блоке EV3

Бывает, что программа, запущенная в «PC-режиме» (кнопкой «Запуск» в Small Basic) прекрасно работает, но при попытке скомпилировать ее в RBF для автономной работы на блоке выходит ошибка. Конечно, что Вы пытаетесь использовать функции, которые не располагает EV3 вроде вывода на экран компьютера, то все понятно, но иногда ошибка не так очевидна. Ниже даны несколько рекомендаций по этому поводу.

- Не используйте функции GraphicWindows, TextWindow и т.п. Вместо функции Timer используйте Program.Delay()
- Small Basic позволяет использовать переменные до того, как им были присвоены значения, например следующий код будет прекрасно работать в PC-режиме, но компилятор EV3 Explorer выдаст ошибку.

```
Sub Move
  Motor.Move("BC",20,angle,"True")
```

```
EndSub
angle=360
Move()
```

Для совместимости с компилятором нужно присвоить переменной значении до ее использования:

```
angle=360
Sub Move
  Motor.Move("BC",20,angle,"True")
EndSub
Move()
```

- Размещайте подпрограммы перед главной программой
- Не используйте следующий способ присвоения значений элементам массива **Colors="0=UNKNOWN;1=BLACK;2=BLUE;3=GREEN"**
- Не используйте строки в качестве индексов массива
- В имени файла .sb не используйте менее 25 символов из разрешенных:
  ABCDEFGHIJKLMNOPQRSTUVWXYZ
  abcdefghijklmnopqrstuvwxyz
  0123456789 -_ (пробелы разрешены)
- В Small Basic не задается тип переменных, однако в EV3 каждая переменная имеет свой тип:

  Число (с 32-битной точностью)
  Тестовая строка
  Массив чисел
  Массив тестовых строк

  Вследствие этих различий переменная будет иметь тип по типу первого присвоенного ей значения

- Авторы перевода обнаружили ошибку компилятора при присваивании переменной очень маленького числа, например **A = 0,0000001** работает в PC-режиме, но не компилируется EV3 Explorer.
