Программируем робота EV3 на языке Python в ev3dev. Краткое руководство и примеры программ

Андрей Степанов
«Карандаш и Самоделкин»

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

1. Экран

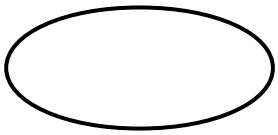
EV3 имеет черно-белый экран с разрешением 178 x 128 пикселей. Левый верхний угол экран имеет координаты (0,0), правый нижний (178,128).

Для работы с экраном используется стандартная Python-библиотека **Pillow**. Далее - несколько методов из нее, если их не достаточно, можно обратиться к документации Pillow.

Нарисовать эллипс (круг) в прямоугольнике, ограниченном координатами
`ellipse(xy, fill=None, outline=None)`

пример

```
from ev3dev.ev3 import *
lcd = Screen()
lcd.draw.ellipse(( 20, 20,  60, 60))
```



Нарисовать прямоугольник (квадрат), ограниченный координатами
`rectangle(xy, fill=None, outline=None)`

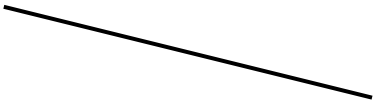
пример

```
from ev3dev.ev3 import *
lcd = Screen()
lcd.draw.rectangle((0,0,177,40), fill='black')
```

Нарисовать линию, из координат в координаты
`line(xy, fill=None, width=0)`

пример

```
from ev3dev.ev3 import *
lcd = Screen()
lcd.draw.line(( 0, 0,  100, 50))
```



Нарисовать точку в заданных координатах
`point(xy, fill=None)`

пример

```
from ev3dev.ev3 import *
lcd = Screen()
lcd.draw.line(( 20, 20))
```

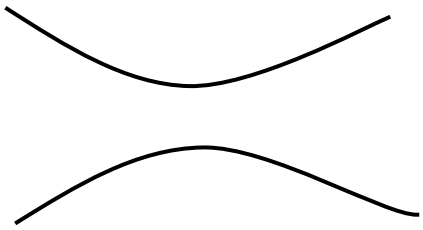
.

Нарисовать кривую, из координат в координаты

```
arc(xy, start, end, fill=None)
```

пример

```
from ev3dev.ev3 import *
lcd = Screen()

lcd.draw.arc((20, 80, 158, 100), 0, 180)

lcd.draw.arc((20, 80, 158, 100), 180, 360)
```



Очистить экран

пример

```
from ev3dev.ev3 import *                              .
lcd = Screen()
lcd.clear()
```

Обновить экран (вывести изображение из буфера)

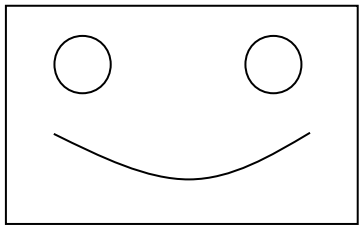
пример

```
from ev3dev.ev3 import *                              .
lcd = Screen()
lcd.update()
```

Следующий пример рисует ан экране смайлик, который раз в секунду меняет настроение с веселого на грустный и наоборот.

пример

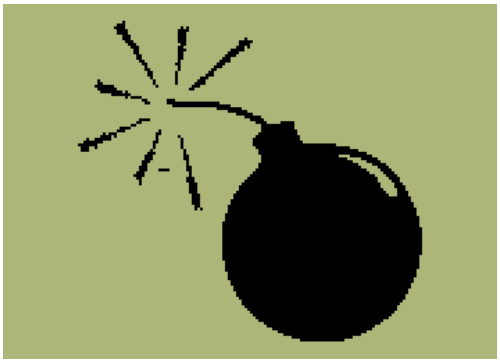
```
from time import sleep
from ev3dev.ev3 import *

lcd = Screen()
smile = True   # сначала смайлик будет веселым

while True:
    lcd.clear()     # очистить экран

    lcd.draw.ellipse(( 20, 20,  60, 60))     # левый глаз
    lcd.draw.ellipse((118, 20, 158, 60))     # правый глаз

    if smile:            # улыбающийся рот
        lcd.draw.arc((20, 80, 158, 100), 0, 180)
    else:                # грустный
        lcd.draw.arc((20, 80, 158, 100), 180, 360)

    smile = not smile    # меняем настроение

    lcd.update() # обновляем экран

    sleep(1)   # ждем секунду
```



Вывод картинки на экран. Поддерживается множество форматов, включая jpg, bmp, gif, png и другие.

пример

```
from ev3dev.ev3 import *
from PIL import Image
```



```
lcd = Screen()

logo = Image.open('pics/Bomb.bmp')
lcd.image.paste(logo, (0,0))
lcd.update()
```

Вывод текста на экран. Следующий пример рисует закрашенный прямоугольник и выводит в нем текст белым. Ниже него выводит текст черным на белом фоне.

пример


```
from ev3dev.ev3 import *
lcd = Screen()

lcd.draw.rectangle((0,0,177,40), fill='black')
lcd.draw.text((48,13),'Hello, world.', fill='white')
lcd.draw.text((36,80),'THIS TEXT IS BLACK')
lcd.update()
```

Функция `textsize()` не изменяет размер текста, как кажется из ее названия. Она определяет размер области в пикселах, которую займет текст.

```
from ev3dev.ev3 import *
lcd = Screen()

my_string='THIS TEXT IS BLACK'
print(lcd.draw.textsize(my_string))
lcd.draw.text((36,80),my_string)
lcd.update()
```

Можно вывести текст произвольным шрифтом. Для начала нужно установить шрифты командой

```
sudo apt-get install ttf-mscorefonts-installer
```

Во время установки блок ev3 должен быть подключен к Интернет. Посмотреть список установившихся шрифтов можно командой

```
ls /usr/share/fonts/truetype/msttcorefonts/
```

Следующий пример выведет шрифтом Arial.ttf во весь экран “Hello world”.

пример


```
from ev3dev.ev3 import *

from PIL import Image, ImageDraw, ImageFont
from time import sleep

lcd = Screen()

f =
ImageFont.truetype('/usr/share/fonts/truetype/msttcorefonts/Aria
l.ttf', 75)
lcd.draw.text((3,0), 'Hello', font=f)
lcd.draw.text((2,55), 'world', font=f)
lcd.update()
```

Дополнительные полезности. Как снять скриншот экрана EV3.

Простейший вариант – ввести в терминале команду **`fbgrab "screenshot.png`**". Скриншот будет сохранен в текущей папке.

Как запустить из терминала программу, которая работает с экраном:

1. Ввести команду **`sudo chvt 6`** Ввести пароль робота (maker по умолчанию). Меню на экране EV3 пропадет .
2. Необязательно, для удобства. Введите **`sudo conspy`** после этого на экран и терминал будет выводиться одна и та же SSH-сессия.
3. Запустить вашу программу
4. Завершить вашу программу - Ctrl+C.
5. Если использовали п.2 – нажмите трижды Esc.
6. Ввести команду **`sudo chvt 1`** чтобы вернуть меню на экране EV3 (программу Brickman)

2. Кнопки на блоке

Блок EV3 имеет 6 кнопок, все можно использовать в программе. Удерживание кнопки “Назад” завершает выполняющуюся программу, если она была запущена из меню блока (из интерфейса Brickman). Далее – несколько примеров

Пример 1

Программа завершается после нажатия любой кнопки на блоке

```
from ev3dev.ev3 import *
from time import sleep

btn = Button()

while True:
    if btn.any():     # Проверяем, если нажата любая кнопка, то
        exit()        # завершаем программу
    else:
        sleep(0.01)   # иначе ждем 0.01 секунду
```

Пример 2

Программа выводит на экран название и состояние кнопки при смене состояния

```
from ev3dev.ev3 import *
from time import sleep

btn = Button()

# Что делать при нажатии или отпускании кнопок

def left(state):
    if state:
        print('Left button pressed')
    else:
        print('Left button released')

def right(state):  # другая, более естественная форма записи
    print('Right button pressed' if state else 'Right button
released')

def up(state):
    print('Up button pressed' if state else 'Up button
released')

def down(state):
    print('Down button pressed' if state else 'Down button
released')

def enter(state):
    print('Enter button pressed' if state else 'Enter button
released')

def backspace(state):
    print('Backspace button pressed' if state else 'Backspace
button released')

btn.on_left = left
btn.on_right = right
btn.on_up = up
btn.on_down = down
btn.on_enter = enter
btn.on_backspace = backspace

while True:  # Бесконечный цикл проверяет состояние кнопок
             # вызывая обработчики при смене состояний
    btn.process() # проверка нажатых кнопок и вызов обработчиков
    sleep(0.01)   # ждем 0.01 секунду
```

Пример 3

Программа решает предыдущую задачу, но более лаконичным способом

```
rom ev3dev.ev3 import *
from time import sleep
```

```
btn = Button()

def change(changed_buttons):   # на вход функции попадает список
# (массив) кнопок, у которых изменилось состояние
    print('These buttons changed state: ' +
str(changed_buttons))

btn.on_change = change

while True:
    btn.process()
    sleep(0.01)
```

Пример 4

Проверка состояния кнопок

```
from ev3dev.ev3 import *
from time import sleep

btn = Button()

while not btn.backspace: # Пока не нажата кнопка НАЗАД
    print(btn.left)      # отображать состояние кнопки влево
    sleep(1)             # с интервалом в одну секунду
```

Пример 5

Вывод списка нажатых кнопок, если нажаты ВЛЕВО+ВПРАВО, то завершить программу

```
from ev3dev.ev3 import *
from time import sleep

btn = Button()

while True:
    print(btn.buttons_pressed)
    if btn.check_buttons(buttons=['left','right']):
        exit()
    sleep(1)
```

3. Подсветка на блоке

EV3 блок имеет подсветку из двух двухцветных светодиодов (красных и зеленых), расположенных под кнопками. Дополнительные цвета получаются их сочетанием в разных пропорциях.

Чтобы изменить цвет подсветки нужно использовать метол **`Leds.set_color(group, color)`**
`Group` в данным методе может принимать значения **`Leds.LEFT`** (левый светодиод) или **`Leds.RIGHT`** (правый светодиод).
`Color` может быть **`RED, GREEN, AMBER, ORANGE`** или **`YELLOW`**.

```
from ev3dev.ev3 import *

# Включаем левый зеленый светодиод
Leds.set_color(Leds.LEFT, Leds.GREEN)
```

Выключить подсветку можно методом **Leds.all_off()**

Python позволяет менять не только цвет подсветки кнопок, но и ее яркость, попробуйте следующий пример – он весьма красиво плавно разжигает и гасит подсветку.

```
from ev3dev.ev3 import *

for i in range(0,255):
    Leds.set_color(Leds.LEFT, Leds.GREEN,i/255)
    Leds.set_color(Leds.RIGHT, Leds.RED,i/255)
for i in range(255,0,-1):
    Leds.set_color(Leds.LEFT, Leds.GREEN,i/255)
    Leds.set_color(Leds.RIGHT, Leds.RED,i/255)
```

4. Динамик

В модулях ev3dev Python есть ряд методов для работы с динамиком.

```
from ev3dev.ev3 import *

# проигрывает стандартный писк
Sound.beep()

# тон с частотой 1500 Гц и длительностью 2 секунды
Sound.tone(1500, 2000)

# последовательность тонов, первый в частотой 3000 Гц
# длительностью 2 секунды, после него пауза 0.4 сек,
# следом второй 800 Гц 1.8 сек, пауза 2 сек
Sound.tone([(3000, 2000, 400),(800, 1800, 2000)])

# воспроизводит wav-файл по указанному пути
Sound.play('sounds/sound01.wav')

# Синтез речи на английском языке
Sound.speak('Hello, my name is E V 3!')
```

По умолчанию при вызове метода, проигрывающего звук программа продолжается и не ждем, пока проигрывание закончится. Чтобы дождаться проигрывания, добавьте **.wait()**

```
Sound.beep().wait()
Sound.tone(440, 500500).wait()
Sound.tone([(400, 1000, 400),(800, 1000, 800)]).wait()
Sound.play('sounds/sound02.wav').wait()
Sound.speak('Hello, world!').wait()
```

II. Использование моторов

При работе с моторами следует учитывать, что скорость моторов **speed_sp** в Python для EV3 измеряется в градусах в секунду и может изменяться в интервале **-900..900.**

1. Поворот мотора на определенный угол.

run_to_rel_pos(position_sp=<угол_градусов>, speed_sp=<скорость>)
Для поворота мотора в обратном направлении необходимо использовать отрицательный угол, а не скорость. Скорость в этом методе всегда в диапазоне 0..900.

Пример

Повернуть большой мотор, подключенный к порту B, на угол 360 градусов и затормозить

```
rom ev3dev.ev3 import *
from time import sleep

m = LargeMotor('outB', stop_action="hold")
m.run_to_rel_pos(position_sp=360, speed_sp=900)
sleep(5)   # Даем мотору время отработать команду
```

Для поворота мотора **в заданную абсолютную позицию** используется метод
run_to_abs_pos(position_sp=<угол_градусов>, speed_sp=<скорость>)

Если к блоку EV3 подключено только по одному мотору каждого типа (**LargeMotor, MediumMotor**), то указывать порт при создании объекта не обязательно, не обязательно, он определится автоматически.

```
m = LargeMotor()
```

2. Поворот мотора в течении заданного промежутка времени:

Пример

Запустить большой мотор подключенный к порту B, со скоростью 675 (675 соответствует 75% в стандартном ПО LEGO) в обратном направлении на 3 секунды.

```
from ev3dev.ev3 import *
from time import sleep

m = LargeMotor('outB')
m.run_timed(time_sp=3000, speed_sp=-675)
print("Скорость в момент старта = " + str(m.speed_sp))
sleep(1)  # За секунду мотор успеет раскрутиться
print("Фактическая скорость через секунду = " + str(m.speed))
sleep(4)
```

За 4+1 секунду ожидания программа не завершится и даст мотору отработать заданные 3 секунды. Для среднего мотора используйте класс **MediumMotor('outB')**

3. Запуск мотора на постоянное вращение

Используйте **run_forever(speed_sp=<скорость>)** для запуска мотора на постоянное вращение. Для остановки мотора используйте **stop()** или **stop(stop_action=<значение>)** где **<значение>** может быть **"coast"** (отключить питание), **"brake"** (пассивное торможение) или **"hold"** (удерживать положение – активное торможение). Параметр **stop_action** можно использовать в других методах, например **run_timed** или **run_to_rel_pos.**

Пример

Запустить большой мотор подключенный к порту B, со скоростью 900. Затормозить через 5 секунд используя активное торможение.

```
from ev3dev.ev3 import *
from time import sleep

m = LargeMotor('outB')
m.run_forever(speed_sp=900)
sleep(5)
m.stop(stop_action="hold")
sleep(5)
```

Запуск мотора без использования регулятора с использованием метода **run-direct(duty_cycle_sp=<значение>)**, где значение изменяется -100..100 отличается тем, что вступает в действие немедленно, в отличии от использующих регулирование методов. Данный метод не контролирует фактическую скорость мотора.

4. Ожидание, пока мотор выполнит команду

Пример

Запустить большой мотор подключенный к порту C, со скоростью 430 на 10 оборотов. Программа завершится когда мотор фактически выполнит эту задачу.

```
from ev3dev.ev3 import *
from time import sleep

m = LargeMotor('outC')
m.run_to_rel_pos(position_sp=3600, speed_sp=430,
stop_action='brake')

while any(m.state):  # не используйте с stop_action='hold'
    sleep(0.1)
Sound.beep().wait()
```

Обратите внимание, что список m.state всегда включает в себя кроме фактически актуального состояния **running** (запущен), **ramping** (разгоняется), **overloaded** (перегружен) и **stalled** (заглох, застопорился) еще и **hold** (удерживается), поэтому использовать этот метод торможения и ожидать выполнения команды вышеуказанным способом нельзя.

5. Дополнительно об остановке моторов

Как вы уже знаете, для остановки мотора используется **stop()** или **stop(stop_action=<значение>)** . Как правило, остановка моторов простым отключением питания (**stop_action="coast")** используется редко, чаще всего применяется пассивное торможение (**stop_action="brake")**, поэтому для удобства при создании объекта мотор удобно сразу указать этот способ торможения по умолчанию:

```
mB=LargeMotor('outB', stop_action='brake')
```

В процессе отладки программ бывает, что после остановки работы программы моторы продолжают вращаться (например программа вылетела с ошибкой). Чтобы остановить работу моторов удобно иметь под рукой скрипт следующего содержания:

```
#!/usr/bin/env python3
from ev3dev.ev3 import *

mB = LargeMotor('outB')
mC = LargeMotor('outC')

mB.stop()
mC.stop()

# и на всякий случай для надежности
mB.run_forever(speed_sp=0)
mC.run_forever(speed_sp=0)
```

6. «Встроенные» регуляторы моторов

Python для EV3 использует встроенные регуляторы для поддержания указанной в **speed_sp** скорости в градусах в секунду. Если нагрузка на мотор возрастает, регулятор автоматически повысит подаваемую мощность, чтобы сохранить указанную скорость и наоборот. Для обхода встроенных регуляторов можно использовать метод **run_direct()**, который соответствует прямому управлению моторами в EV3-G (синие блоки). Напоминем ,в этом методе не используется **speed_sp**, а применяется **duty_cycle_sp**, изменяемая в диапазоне -100..100.

III. Датчики

EV3 Python совместим со всеми стандартными EV3 и NXT датчиками, однако датчик NXT Color Sensor пока не может быть использован для определения цвета, его можно использовать только как датчик освещенности. Также моддерживается большое число сторонних датчиков от Hitechnic, Mindsensors, а также самодельные датчики.

Как и в случае моторов, если вы используете не более одного датчика каждого типа, можете не указывать в программе порт, к которому подключен датчик, он определится автоматически.

Перед началом использования каждого датчика, его нужно инициализировать, указав режим работы датчика. Режимы работы для основных типов EV3 и NXT датчиков приведены в таблицах ниже.

EV3 датчики

Режим	Описание	Ед.изм.	Точность	Значение
TOUCH	Состояние кнопки	-	1	0 или 1
COL-REFLECT COL-AMBIENT COL-COLOR RGB-RAW	Отраженный свет. Подсветка красная Внешняя освещенность. Подсветка синяя Цвет. Подсветка белая Цвет в RAW. Подсветка белая	процент процент цвет -	1 1 1 1	0-100 0-100 0-7* **
US-DIST-CM US-DIST-IN	Расстояние в мм ,Расстояние в десятых дюйма	мм дюйм	.1 .1	0-2550 0-1003
GYRO-ANG GYRO-RATE	Угол Скорость поворота	градус град/сек	1 1	+-32767 +-440
IR-PROX	Расстояние (100% соответствует 70 см/27 дюймам)	процент	1	100

* 0=неизвестно, 1=черный, 2=синий, 3=зеленый, 4=желтый, 5=красный, 6=белый, 7=коричневый

** возвращает 3 значения, каждое в диапазоне 0-1023: **`value(0): Red , value(1): Green, value(2): Blue`**.

NXT датчики

Режим	Описание	Ед.изм.	Точность	Значение
TOUCH	Состояние кнопки	-	1	0 или 1
REFLECT AMBIENT COLOR	Отраженный свет. Подсветка включена Внешняя освещенность. Подсветка выкл. **ПОКА НЕ ПОДДЕРЖИВАЕТСЯ!**	процент процент	.1 .1	0-1000 0-1000
DB	Звуковое давление	процент	.1	0-1000
US-DIST-CM US-DIST-IN	Расстояние в см ,Расстояние в десятых дюйма	см дюйм	0 1	0-255 0-1000

1. Датчик касания (кнопка)

Датчик касания (кнопка) возвращает 0, если кнопка не нажата и 1 в нажатом положении.

Пример

Программа издаст звуковой сигнал и завершится при нажатии датчика-кнопки, подключенного к любому порту.

```
from ev3dev.ev3 import *
```

```
ts = TouchSensor()
# проверка, подключен ли датчик
# программа завершится с сообщением, если датчик не подключен
assert ts.connected, "Подключите датчик касания в любой порт"

while not ts.value():     # Остановить программу при нажатии
    touch = ts.value()

Sound.beep()
```

Пример

В этом примере используется два датчика-кнопки, подлеченных к 1 и 2 портам. При нажатии первой кнопки левый светодиод подсветки меняет цвет с зеленого на красный, при нажатии второй тоже самое происходит в правым светодиодом. Программа завершится, если нажать обе кнопки одновременно.

```
from ev3dev.ev3 import *
from time import sleep

ts1 = TouchSensor('in1')
assert ts1.connected, "Подключите датчик-кнопку в порт 1"

ts2 = TouchSensor('in2')
assert ts2.connected, "Подключите датчик-кнопку в порт 2"

while not ts1.value() and not ts2.value():
    Leds.set_color(Leds.LEFT, (Leds.GREEN, Leds.RED)
[ts1.value()])
    Leds.set_color(Leds.RIGHT, (Leds.GREEN, Leds.RED)
[ts2.value()])
```

2. Цветосветовой датчик в режиме измерения освещенности

Пример

Вывод в консоль показаний датчика цвета EV4 в режиме измерения освещенности.

```
from ev3dev.ev3 import *
from time import sleep

cl = ColorSensor()
assert cl.connected, "Подключите датчик цвета EV3 в любой порт"

# Переводим датчик в режим измерения освещенности
# в этом режиме датчик выдает освещенность 0..100%
cl.mode='COL-REFLECT'

while True:
    print(cl.value())
    sleep(0.5)
```

3. Цветосветовой датчик в режиме измерения отраженного света

Пример

Скорость большого мотора будет пропорциональна освещенности на датчике цвета.

```
from ev3dev.ev3 import *
from time import sleep

cl = ColorSensor()
assert cl.connected, "Подключите датчик цвета EV3 в любой порт"

m = LargeMotor('outB')
assert m.connected, "Подключите большой мотор в порт B"

# Переключим датчик цвета EV3
# в режим измерения отраженного света
# в этом режиме датчик выдает освещенность 0..100%
cl.mode='COL-AMBIENT'

# run_forever метод позволяет реругировать скорость
# путем изменения «на лету» значения speed_sp
m.run_forever(speed_sp = 0)

while True:
# скорость мотора будет пропорциональна освещенности
    m.speed_sp = cl.value()
```

4. Цветосветовой датчик в режиме измерения цвета

Пример

Робот произносит названия цветов, полученные в результате определения датчиком.

```
from ev3dev.ev3 import *
from time import sleep

cl = ColorSensor()
assert cl.connected, "Подключите датчик цвета EV3 в любой порт"

# Переключим датчик цвета EV3 в режим измерения цвета
cl.mode='COL-COLOR'

colors=('unknown','black','blue','green','yellow','red','white',
'brown')
while True:
    Sound.speak(colors[cl.value()]).wait()
    sleep(1)
```

5. Цветосветовой датчик в режиме измерения RGB-RAW

Этот режим является особенным, поскольку в нем датчик возвращает сразу три значения – количество отраженного света в красной, зеленой и синей его составляющей. Любой цвет можно представить как смесь этих трех цветов в определенной пропорции.

Пример

Вывод на консоль значения красной, зеленой и синей составляющей текущего цвета.

```
from ev3dev.ev3 import *
from time import sleep

cl = ColorSensor()
assert cl.connected, "Подключите датчик цвета EV3 в любой порт"

# Переключим датчик цвета EV3 в режим RGB-RAW
cl.mode='RGB-RAW'

while True:
    red = cl.value(0)
    green=cl.value(1)
    blue=cl.value(2)
    print("Red: " + str(red) + ", Green: " + str(green) + ",
Blue: " + str(blue))
    sleep(1)
```

6. Ультразвуковой датчик

Используя ультразвуковой датчик EV3 в Python, следует учитывать, что датчик возвращает данные в миллиметрах и в десятых долях дюйма соответственно, поэтому, чтобы получить сантиметры и дюймы необходимо разделить полученные значение на 10. Это замечание не касается ультразвукового датчика NXT, который возвращает показания в сантиметрах.

Пример

Программа два раза в секунду выводит на консоль показания ультразвукового датчика в сантиметрах и меняет цвет подсветки в зависимости от его величины.

```
from ev3dev.ev3 import *
from time import sleep

us = UltrasonicSensor()
assert us.connected, "Подключите УЗ-датчик EV3 в любой порт"

# Переводим УЗ датчик в режим измерения в ММ
us.mode='US-DIST-CM'

units = us.units
# вернет СМ, хотя датчик будет измерять в ММ

while True:
    distance = us.value()/10  # Переводим ММ в СМ
    print(str(distance) + " " + units)
```

```
    if distance < 60:
        Leds.set_color(Leds.LEFT, Leds.RED)
    else:
        Leds.set_color(Leds.LEFT, Leds.GREEN)

    sleep(0.5)
```

7. Инфракрасный датчик в режиме измерения расстояния

Пример

Программа меняет цвет подсветки в зависимости от показаний ИК-дальномера.

```
from ev3dev.ev3 import *

ir = InfraredSensor()
assert ir.connected, "Подключите ИК-датчик EV3 в любой порт"

# Устанавливаем ИК-датчик в режим измерения расстояния
ir.mode = 'IR-PROX'

while True:
    # ИК-датчик только приблизительно дает представление
    # об удаленности объекта
    # чтобы сконвертировать в см, умножим показания на 0.7

    distance = 0.7 * ir.value()

    if distance < 50:
        Leds.set_color(Leds.LEFT, Leds.RED)
    else:
        Leds.set_color(Leds.LEFT, Leds.GREEN)
```

1. Инфракрасный датчик и ИК-маяк в режиме пульта ДУ

Пример

Программа управляет двумя моторами, подключенными к портам B и C, отслеживая нажатия кнопок на пульте ДУ. Кроме этого меняется цвет подсветки кнопок в зависимости от нажатых на пульте кнопок.

```
from ev3dev.ev3 import *
from time import sleep

# Подключите пару больших моторов к портам B и C
lmotor = LargeMotor('outB')
rmotor = LargeMotor('outC')

# Проверяем что не забыли подключить моторы
assert lmotor.connected, "Левый мотор не подключен"
assert rmotor.connected, "Правый мотор не подключен"
```

```
# Подличаем пульт ДУ
# Точка с запятой позволяет записать несколько команда в строке
rc = RemoteControl(); assert rc.connected

# Выключаем подсветку кнопок блока EV3
Leds.all_off()

def roll(motor, led_group, direction):
    """
    Многострочный комментарий в Python обрамляется тремя
кавычками.
    Здесь мы создаем обработчик событий.
    Функция on_press необходима для класса RemoteControl.
    Она принимает значение True, когда кнопка нажата, False –
когда отпущена.
    """
    def on_press(state):
        if state:
            # Если кнопка нажата, включаем мотор в направлении
            # direction и включаем подсветку кнопок
            motor.run_forever(speed_sp=90*direction)
            Leds.set_color(led_group, direction > 0 and
Leds.GREEN or Leds.RED)
        else:
            # Если отпущена, выключаем моторы, гасим подсветку
            motor.stop(stop_action='brake')
            Leds.set(led_group, brightness_pct=0)

    return on_press

# Назначаем обработчики прерываний на кнопки пульта
rc.on_red_up    = roll(lmotor, Leds.LEFT,   1)
rc.on_red_down  = roll(lmotor, Leds.LEFT,  -1)
rc.on_blue_up   = roll(rmotor, Leds.RIGHT,  1)
rc.on_blue_down = roll(rmotor, Leds.RIGHT, -1)

# Запускаем цикл, в котором контролируем нажатые кнопки
while True:
    rc.process()
    sleep(0.01)
```

2. Гироскоп

Обратите внимание - датчик гироскопа в момент запуска программы должен быть абсолютно неподвижен!

Пример

Программа два раза в секунду выводит на консоль показания гироскопа в градусах и воспроизводит звук, зависящий от текущего угла.

```
from ev3dev.ev3 import *
from time import sleep
```

```
gy = GyroSensor()
assert gy.connected, "Подключите гироскоп EV3 в любой порт"

# Переключаем гироскоп в режим измерения угла
gy.mode='GYRO-ANG'

units = gy.units
# Вернет 'deg' в режиме измерения угла

while True:
    angle = gy.value()
    print(str(angle) + " " + units)
    Sound.tone(1000+angle*10, 500).wait()
    sleep(0.5)
```

3. Мотор как датчик угла

Пример

Запустить большой мотор подключенный к порту B, со скоростью 900 град/сек на один оборот, вывести в консоль показания энкодера, сбросить энкодер, повторно вывести показания.

```
rom ev3dev.ev3 import *
from time import sleep

m = LargeMotor('outB', stop_action="hold")
m.run_to_rel_pos(position_sp=360, speed_sp=900)
sleep(5)   # Даем мотору время отработать команду
encB = m.position
print (encB)
m.reset()  # Эта команда также останавливает мотор
sleep(1)
encB = m.position
print (encB)
```

IV. Другие полезные функции

1. Энергопитание

```
from ev3dev.ev3 import *

battery = PowerSupply()
voltage = battery.measured_voltage
current = battery.measured_current
amps = battery.measured_amps
volts = battery.measured_volts
print(voltage)
```

2. Случайные числа

Пример

Использование г генератора случайных чисел.

```
import random

print (random.randint(-100,100)
print (random.random())
```

3. Математические функции

abs(x) – возвращает абсолютное значение числа (модуль), т.е. abs(-36.6) = 36.6
round(number) - округляет число до ближайшего целого
round(number, ndigits) – округляет число с плавающей точкой до указанного во втором параметре числа знаков после запятой, пример round(2.78, 1) = 2.8

Для функций abs() и round() нет необходимости подключать библиотеку math.

Тригонометрические функции требуют подключения модуля math. Тригонометрические функции в Python работают с радианами, а не градусами. Сконвертировать градусы в радианы можно функцией **math.radians(x)**, а радианы в градусы функцией **math.degrees(x)**.

Пример

Вывести значение косинуса угла в 45 градусов на консоль.

```
import math

print (math.cos(math.radians(45)))
```

4. Таймер, работа со временем

Пример

Измерим длительность стандартного звука beep().

```
from time import sleep, time

starttime = time()
Sound.beep()
alltime = time() - starttime
print (alltime)
sleep(4)
```

5. Параллельные потоки (задачи)

Пример запуска параллельного потока

```
import threading
from time import sleep, time

def my_timer_task():
    """параллельный поток"""
    global my_timer
    while True:
        my_timer = round(time() - timer_start)
        sleep(1)


timer_start = time()
my_timer = time() - timer_start

# запуск параллельной задачи
t = threading.Thread(target=my_timer_task)
t.daemon = True
t.start()

# вывод данных, получаемых из параллельной задачи
while True:
     print(my_timer)
     sleep(1)
```

FAQ

Если программа не запускается из Brickman

Программа, написанная в редакторе под Windows, может содержать переводы строк, не допустимые в Linux. При этом программа прекрасно запускается из консоли, но вылетает, будучи запущенной из меню блока EV3. В этом случае нужно удалить переводы строк Windows, воспользовавшись командой в консоли:

```
sed -i 's/\r//g' <file>
```