A POSTERIORI

Play · Experience · Learn

# PYTHON COMMANDS ON EV3DEV

## Introduction

The full list of commands (...also known as Application Programming Interface “API”) for Python on EV3DEV can be found here:

http://python-ev3dev.readthedocs.io/en/stable/spec.html

As the full documentation can be overwhelming for students new to Python, we have selected some commonly used commands and simplified them here. This will give the reader an easier start to programing the EV3 using Python, but please be aware that not all commands are listed here and those that are listed are often simplified. Students should try reading the full documentation as soon as they are comfortable with Python.

## Motors

While there are separate classes for LargeMotor and MediumMotor, both of them works in the same way.

### INITIALIZE A NEW INSTANCE

You need to have this command before anything else. I use the name “m” in this document, but you can choose any name that you like.

```
m = LargeMotor(<address>)
m = MediumMotor(<address>)
```

<address> : May be any of the following **outA**, **outB**, **outC**, **outD**

**Example:**

```
m = LargeMotor('outA')
```

### SET SPEED

Maximum speed is **1050** for a large motor and **1560** for a medium motor. The unit is in degrees per second and a negative value will cause the motor to run in reverse. Note that this is only a setting; the motor will only run when a command is given.

**Example:**

```
m.speed_sp = 1000
```

## SET RUN DURATION

This specifies the duration that the motor will run when the ***run-timed*** command is used. The unit is in milliseconds. Note that this is only a setting; the motor will only run when a command is given.

**Example:**

```
m.time_sp = 1000
```

## SET TARGET POSITION

This specifies the position that the motor will run to when the ***run_to_rel_pos*** command is used. The unit is in degrees. Note that this is only a setting; the motor will only run when a command is given.

**Example:**

```
m.position_sp = 100
```

## SET STOP ACTION

This specifies how the motor will stop. Valid values are **coast**, **brake**, and **hold**.

**coast:** Motor will freely coast to a stop. Robot may travel some distance before stopping.

**brake:** Motor will be given a passive load to stop it. Compared to coast, the robot will travel a shorter distance before stopping.

**hold:** Motor will consume power to try and stop immediately. The robot will stop the fastest in this mode.

**Example:**

```
m.stop_action = 'coast'
```

## RUN MOTOR FOREVER

This will run the motor until another run command or a stop command is given. Note that you need to set ***speed_sp*** before running this command.

**Example:**

```
m.speed_sp = 1000
m.run_forever()
```

The above commands will run the motor continuously at a speed of 1000 degrees per second.

## RUN MOTOR FOR DURATION

This will run the motor for the duration specified in ***time_sp***. Note that you need to set ***speed_sp*** and ***time_sp*** before running this command.

**Example:**

```
m.speed_sp = 1000
m.time_sp = 500
m.run_timed()
```

The above commands will run the motor for 500 milliseconds at a speed of 1000 degrees per second.

## RUN MOTOR TO RELATIVE POSITION

This will run the motor to the specified position. Note that you need to set ***speed_sp*** and ***position_sp*** before running this command.

**Example:**

```
m.speed_sp = 1000
m.position_sp = 800
m.run_to_rel_pos()
```

The above commands will turn the motor 800 degrees at a speed of 1000 degrees per second.

## WAIT UNTIL NOT MOVING

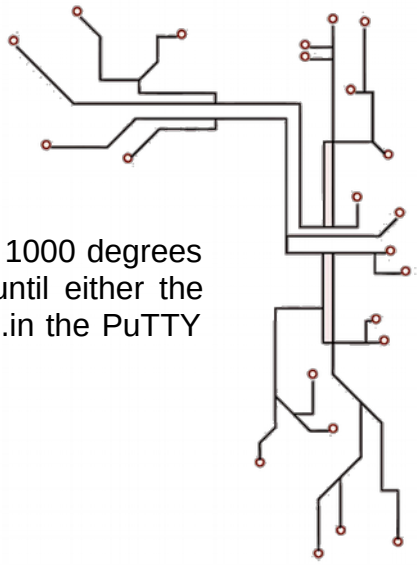
When issuing a “Move Rotation” or “Move Duration” command in the original Lego EV3 software, the program stops and wait for the move to complete before continuing. In Python, the program does not stop to wait for any move to complete. If you want your program to stop and wait for a move to complete, you should use this command.

```
wait_until_not_moving(<timeout>)
```

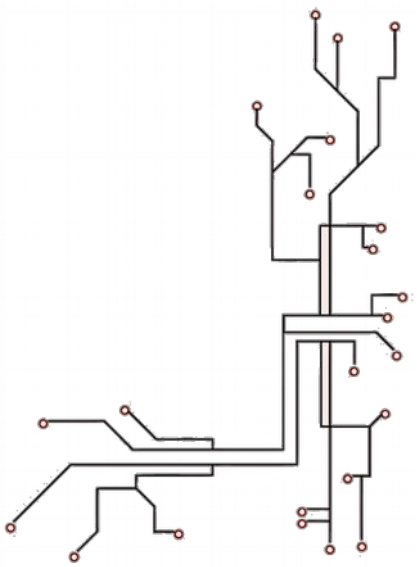
<timeout> : This command will stop the program from continuing, until either the motor stops moving, or until the timeout is reached. Unit is in milliseconds.

**Example:**

```
m.speed_sp = 1000
m.position_sp = 8000
m.run_to_rel_pos()
m.wait_until_not_moving(3000)
print(‘Foo’)
```



The above commands will turn the motor 8000 degrees at a speed of 1000 degrees per second. The program will stop at ***wait_until_not_moving*** line until either the motor stops or 3000 milliseconds is up, then display the word ‘Foo’ (...in the PuTTY screen, not the EV3 screen).



# Color Sensor

## INITIALIZE A NEW INSTANCE

You need to have this command before anything else. I use the name “s” in this document, but you can choose any name that you like.

```
s = ColorSensor(<address>)
```

<address> : May be any of the following **in1**, **in2**, **in3**, **in4**

**Example:**

```
s = ColorSensor('in1')
```

## READ REFLECTED LIGHT

Provides the reflected light intensity. The unit is in percentage (0 to 100).

**Example:**

```
print(s.reflected_light_intensity)
```

`The above command will read the reflected light intensity and print it to screen` (...in the PuTTY screen, not the EV3 screen).

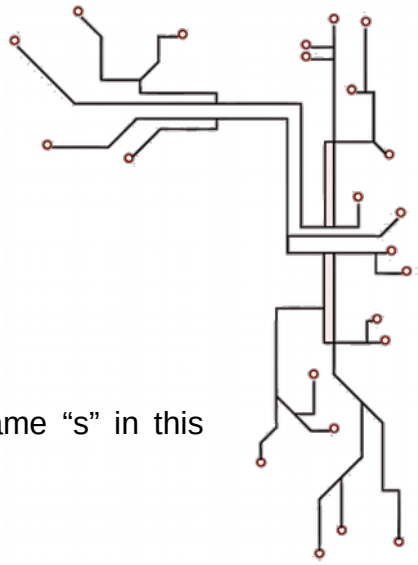
## READ RED / GREEN / BLUE

Provides the detected light intensity for red, green, or blue. The value ranges from 0 to 1020.

**Example:**

```
print(s.red)
print(s.green)
print(s.blue)
```

The above command will read the amount of red / green / blue light detected and print it to screen (...in the PuTTY screen, not the EV3 screen).



# Touch Sensor

## INITIALIZE A NEW INSTANCE

You need to have this command before anything else. I use the name “s” in this document, but you can choose any name that you like.

```
s = TouchSensor(<address>)
```

<address> : May be any of the following **in1**, **in2**, **in3**, **in4**

**Example:**

```
s = TouchSensor('in1')
```

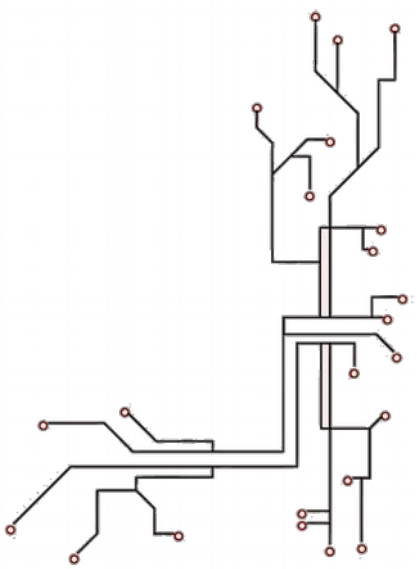
## READ BUTTON PRESS STATE

Have the value “True” if the button is pressed, and “False” if it is not.

**Example:**

```
print(s.is_pressed)
```

The above command will read the button status and print it to screen (...in the PuTTY screen, not the EV3 screen).



# Ultrasonic Sensor

## INITIALIZE A NEW INSTANCE

You need to have this command before anything else. I use the name “s” in this document, but you can choose any name that you like.

```
s = UltrasonicSensor(<address>)
```

<address> : May be any of the following **in1**, **in2**, **in3**, **in4**

**Example:**

```
s = UltrasonicSensor('in1')
```

## READ DISTANCE

Provides the detected distance. The unit is in centimeters.

**Example:**

```
print(s.distance_centimeters)
```

The above command will read the distance from the ultrasonic sensor and print it to screen (...in the PuTTY screen, not the EV3 screen).

# Screen

When drawing to screen, the EV3DEV library uses functions from the Python Imaging Library (PIL). Read the PIL documents (http://effbot.org/imagingbook/pil-index.htm) if you wish to perform advanced screen drawing. In this document, I will only be covering basic text display.

**IMPORTANT:** When running on the command line (via SSH), Brickman (...the software interface that you see when you boot up the EV3) will interfere with the screen display. Programs that are selected and ran using the EV3 will not have this issue.

## INITIALIZE A NEW INSTANCE

You need to have this command before anything else. I use the name “scr” in this document, but you can choose any name that you like.

**Example:**

```
scr = Screen()
```

## CLEAR SCREEN

Clears the screen.

**IMPORTANT:** Changes to the screen will only take effect when the update function is called.

**Example:**

```
scr.clear()
scr.update()
```

Without the second line, you will not see any changes to the screen.

## DRAW TEXT

Draws text to the screen. The ***draw*** function provides a handler to the ***PIL.ImageDraw.Draw*** class. The ***text*** function belongs to PIL.

**IMPORTANT:** Changes to the screen will only take effect when the update function is called.

```
scr.draw.text((<x>,<y>), <text>, <font>)
```

<x> : X coordinate to draw text. From 0 to 178.
<y> : Y coordinate to draw text. From 0 to 128.
<text> : String to draw. If drawing a number, use ***str()*** to convert it to string.

<font> : Optional. Font to use. See example 2.

**Example 1:**

```
scr.clear()
scr.draw.text((0,10), 'Foobar ' + str(123))
scr.update()
```

This draws the text "Foobar 123" on the EV3's screen. Note the use of ***str()*** to convert the number 123 into a string.

**Example 2:**

```
import ev3dev.fonts as fonts
scr.clear()
scr.draw.text((0,10), 'Foo', font=fonts.load('luBS14'))
scr.update()
```

The first line loads a set of fonts that are suitable for the EV3 screen. In the third line, we added a "font" parameter, and specified the "luBS14" font. See http://python-ev3dev.readthedocs.io/en/stable/other.html#screen for a complete list of available fonts.

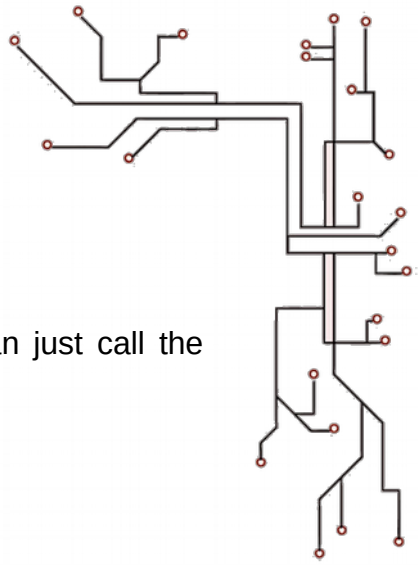
## UPDATE SCREEN

Update all clears and draws to the screen. Without this function, there will be no changes to the screen.

**Example:**

```
scr.clear()
scr.draw.text((0,10), 'Foobar')
scr.update()
```

Clears the screen, draw the word "Foobar", and display all the changes on screen.



# LED

You do not need to initialize an instance to control the LED. You can just call the functions directly.

## OFF LEDS

Turn off all LEDs.

**Example:**

```
Leds.all_off()
```

The above will turn off all LEDs.

## SET LED COLORS

The EV3 has two LEDs, left and right, behind the buttons. This function allows you to set them to various pre-defined colors (example 1), or your own custom color (example 2).

```
Leds.set_color(<position>, <color>)
```

<position> : Either “Leds.LEFT” or “Leds.RIGHT”
<color> : Either a pre-defined color or a truple in the format (red, green) where red and green can be any number from 0 (fully off) to 1 (fully on).

**Pre-defined Colors:**

Leds.RED
Leds.GREEN
Leds.AMBER
Leds.ORANGE
Leds.YELLOW.

**Example 1:**

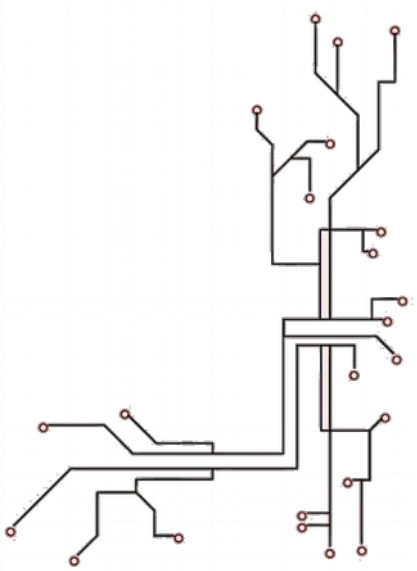
```
Leds.set_color(Leds.LEFT, Leds.RED)
```

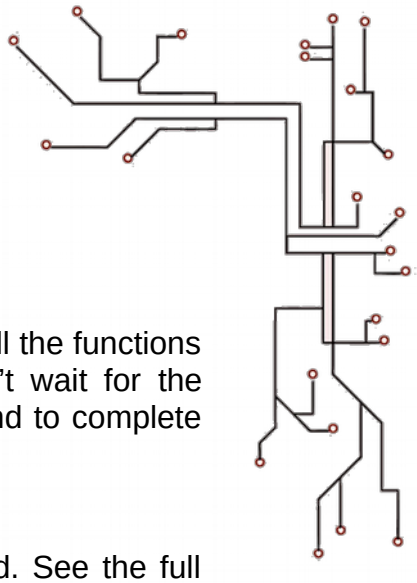
Set the left LED to red color.

**Example 2:**

```
Leds.set_color(Leds.RIGHT, (0.8, 0.5))
```

Set the right LED to 80% red and 50% green.





# SOUND

You do not need to initialize an instance to play sound. You can just call the functions directly. Note that all sound functions are non-blocking (...they don't wait for the sound to finish playing). If you want your program to wait for the sound to complete before continuing, add a ***.wait()*** to the end of the command.

## PLAY TONE

Play a tone. Note that there is an alternative format to this command. See the full documentation for details.

```
Sound.tone(<frequency>, <duration>)
```

<frequency> : Frequency to play.
<duration> : Duration to play in milliseconds.

**Example:**

```
Sound.tone(1000, 500)
```

Play a 1000Hz tone for half a second.

## SPEAK TEXT

Convert a text to audio and play it.

```
Sound.speak(<text>)
```

<text> : Text to speak. If drawing a number, use ***str()*** to convert it to string.

**Example:**

```
Sound.speak('The answer is ' + str(42)).wait()
```

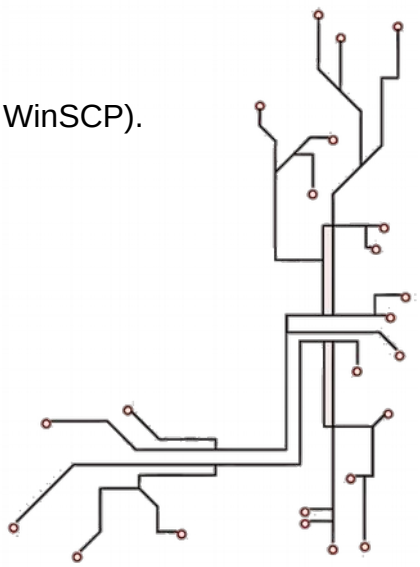
This makes the robot say "The answer is 42". Note the use of ***str()*** to convert the number 42 into a string. The ***.wait()*** at the end will cause the program to pause until the audio finish playing.

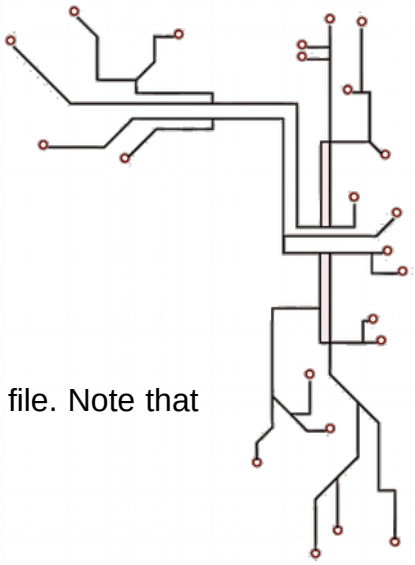
## PLAY WAV FILE

Play a wav file. You'll need to upload the file to the EV3 (...you can use WinSCP).

```
Sound.play(<file>)
```

<file> : Filename of the wav file.





**Example:**

```
Sound.play('meow.wav')
```

Play the file “meow.wav”. This will only work if you first upload the wav file. Note that the file must be in wav format.