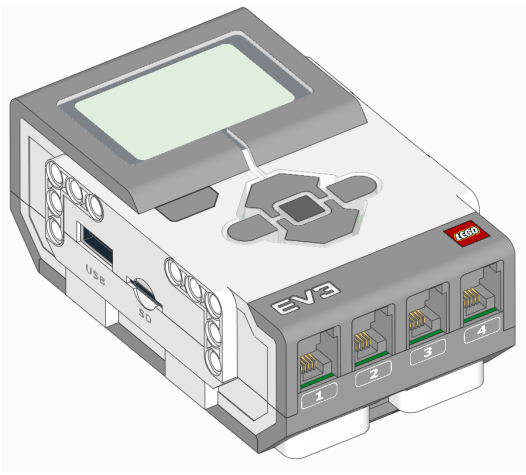
hubs – Programmable Hubs



class **EV3Brick**

LEGO® MINDSTORMS® EV3 Brick.

Using the buttons

buttons.pressed()

Checks which buttons are currently pressed.

Returns: List of pressed buttons.

Return type: List of `Button`

Using the brick status light

light.on(*color*)

Turns on the light at the specified color.

Parameters: **color** (*Color*) – Color of the light. The light turns off if you choose `None` or a color that is not available.

Show/hide example ▲

Example: Turn the light on and change the color.

```
#!/usr/bin/env pybricks-micropython

from pybricks.hubs import EV3Brick
from pybricks.tools import wait
from pybricks.parameters import Color

# Initialize the EV3
ev3 = EV3Brick()

# Turn on a red light
ev3.light.on(Color.RED)

# Wait
wait(1000)

# Turn the light off
ev3.light.off()
```

light.off()

Turns off the light.

Using the speaker

speaker.beep(*frequency=500*, *duration=100*)

Play a beep/tone.

Parameters:

- **frequency** (frequency: Hz) – Frequency of the beep. Frequencies below 100 are treated as 100.
- **duration** (time: ms) – Duration of the beep. If the duration is less than 0, then the method returns immediately and the frequency play continues to play indefinitely.

speaker.play_notes(*notes*, *tempo=120*)

Plays a sequence of musical notes.

For example, you can play: `['C4/4', 'C4/4', 'G4/4', 'G4/4']`.

Parameters:

- **notes** (*iter*) – A sequence of notes to be played (see format below).
- **tempo** (*int*) – Beats per minute where a quarter note is one beat.

Show/hide musical note format ▲

Each note is a string with the following format:

- The first character is the name of the note, `A` to `G` or `R` for a rest.
- Note names can also include an accidental `#` (sharp) or `b` (flat). `B#` / `Cb` and `E#` / `Fb` are not allowed.
- The note name is followed by the octave number `2` to `8`. For example `C4` is middle C. The octave changes to the next number at the note C, for example, `B3` is the note below middle C (`C4`).
- The octave is followed by `/` and a number that indicates the size of the note. For example `/4` is a quarter note, `/8` is an eighth note and so on.
- This can optionally followed by a `.` to make a dotted note. Dotted notes are 1-1/2 times as long as notes without a dot.
- The note can optionally end with a `_` which is a tie or a slur. This causes there to be no pause between this note and the next note.

`speaker.play_file`(*file_name*)

Plays a sound file.

Parameters: **file_name** (*str*) – Path to the sound file, including the file extension.

`speaker.say`(*text*)

Says a given text string.

You can configure the language and voice of the text using `set_speech_options()`.

Parameters: **text** (*str*) – What to say.

`speaker.set_speech_options`(*language=None*, *voice=None*, *speed=None*, *pitch=None*)

Configures speech settings used by the `say()` method.

Any option that is set to `None` will not be changed. If an option is set to an invalid value `say()` will use the default value instead.

Parameters:

- **language** (*str*) – Language of the text. For example, you can choose `'en'` (English) or `'de'` (German). A list of all available languages is given below.
- **voice** (*str*) – The voice to use. For example, you can choose `'f1'` (female voice variant 1) or `'m3'` (male voice variant 3). A list of all available voices is given below.
- **speed** (*int*) – Number of words per minute.
- **pitch** (*int*) – Pitch (0 to 99). Higher numbers make the voice higher pitched and lower numbers make the voice lower pitched.

Show/hide available languages and voices ▲

You can choose the following languages:

- `'af'` : Afrikaans
- `'an'` : Aragonese
- `'bg'` : Bulgarian
- `'bs'` : Bosnian
- `'ca'` : Catalan
- `'cs'` : Czech
- `'cy'` : Welsh
- `'da'` : Danish
- `'de'` : German
- `'el'` : Greek
- `'en'` : English (default)
- `'en-gb'` : English (United Kingdom)
- `'en-sc'` : English (Scotland)
- `'en-uk-north'` : English (United Kingdom, Northern)
- `'en-uk-rp'` : English (United Kingdom, Received Pronunciation)
- `'en-uk-wmids'` : English (United Kingdom, West Midlands)
- `'en-us'` : English (United States)
- `'en-wi'` : English (West Indies)
- `'eo'` : Esperanto
- `'es'` : Spanish
- `'es-la'` : Spanish (Latin America)
- `'et'` : Estonian
- `'fa'` : Persian
- `'fa-pin'` : Persian
- `'fi'` : Finnish
- `'fr-be'` : French (Belgium)
- `'fr-fr'` : French (France)
- `'ga'` : Irish
- `'grc'` : Greek
- `'hi'` : Hindi
- `'hr'` : Croatian
- `'hu'` : Hungarian
- `'hy'` : Armenian
- `'hy-west'` : Armenian (Western)
- `'id'` : Indonesian
- `'is'` : Icelandic
- `'it'` : Italian
- `'jbo'` : Lojban
- `'ka'` : Georgian
- `'kn'` : Kannada
- `'ku'` : Kurdish
- `'la'` : Latin
- `'lfn'` : Lingua Franca Nova

- `'lt'`: Lithuanian
- `'lv'`: Latvian
- `'mk'`: Macedonian
- `'ml'`: Malayalam
- `'ms'`: Malay
- `'ne'`: Nepali
- `'nl'`: Dutch
- `'no'`: Norwegian
- `'pa'`: Punjabi
- `'pl'`: Polish
- `'pt-br'`: Portuguese (Brazil)
- `'pt-pt'`: Portuguese (Portugal)
- `'ro'`: Romanian
- `'ru'`: Russian
- `'sk'`: Slovak
- `'sq'`: Albanian
- `'sr'`: Serbian
- `'sv'`: Swedish
- `'sw'`: Swahili
- `'ta'`: Tamil
- `'tr'`: Turkish
- `'vi'`: Vietnamese
- `'vi-hue'`: Vietnamese (Hue)
- `'vi-sgn'`: Vietnamese (Saigon)
- `'zh'`: Mandarin Chinese
- `'zh-yue'`: Cantonese Chinese

You can choose the following voices:

- `'f1'`: female variant 1
- `'f2'`: female variant 2
- `'f3'`: female variant 3
- `'f4'`: female variant 4
- `'f5'`: female variant 5
- `'m1'`: male variant 1
- `'m2'`: male variant 2
- `'m3'`: male variant 3
- `'m4'`: male variant 4
- `'m5'`: male variant 5
- `'m6'`: male variant 6
- `'m7'`: male variant 7
- `'croak'`: croak
- `'whisper'`: whisper
- `'whisperf'`: female whisper

speaker.set_volume(*volume*, *which='_all_'*)

Sets the speaker volume.

Parameters:

- **volume** (percentage: %) – Volume of the speaker.
- **which** (*str*) – Which volume to set. `'Beep'` sets the volume for `beep()` and `play_notes()`. `'PCM'` sets the volume for `play_file()` and `say()`. `'_all_'` sets both at the same time.

Using the screen

screen.clear()

Clears the screen. All pixels on the screen will be set to `Color.WHITE`.

screen.draw_text(*x*, *y*, *text*, *text_color=Color.BLACK*, *background_color=None*)

Draws text on the screen.

The most recent font set using `set_font()` will be used or `Font.DEFAULT` if no font has been set yet.

Parameters:

- **x** (*int*) – The x-axis value where the left side of the text will start.
- **y** (*int*) – The y-axis value where the top of the text will start.
- **text** (*str*) – The text to draw.
- **text_color** (*Color*) – The color used for drawing the text.
- **background_color** (*Color*) – The color used to fill the rectangle behind the text or `None` for transparent background.

screen.print(**args*, *sep=' '*, *end='\n'*)

Prints a line of text on the screen.

This method works like the builtin `print()` function, but it writes on the screen instead.

You can set the font using `set_font()`. If no font has been set, `Font.DEFAULT` will be used. The text is always printed used black text with a white background.

Unlike the builtin `print()`, the text does not wrap if it is too wide to fit on the screen. It just gets cut off. But if the text would go off of the bottom of the screen, the entire image is scrolled up and the text is printed in the new blank area at the bottom of the screen.

Parameters:

- * (*object*) – Zero or more objects to print.
- **sep** (*str*) – Separator that will be placed between each object that is printed.
- **end** (*str*) – End of line that will be printed after the last object.

Show/hide example ▲

Example: Say hello... in several ways.

```
#!/usr/bin/env pybricks-micropython

from pybricks.hubs import EV3Brick
from pybricks.tools import wait
from pybricks.media.ev3dev import Font

# It takes some time for fonts to load from file, so it is best to only
# load them once at the beginning of the program like this:
tiny_font = Font(size=6)
big_font = Font(size=24, bold=True)
chinese_font = Font(size=24, lang='zh-cn')


# Initialize the EV3
ev3 = EV3Brick()


# Say hello
ev3.screen.print('Hello!')

# Say tiny hello
ev3.screen.set_font(tiny_font)
ev3.screen.print('hello')

# Say big hello
ev3.screen.set_font(big_font)
ev3.screen.print('HELLO')

# Say Chinese hello
ev3.screen.set_font(chinese_font)
ev3.screen.print('你好')

# Wait some time to look at the screen
wait(5000)
```

`screen.set_font`(*font*)

Sets the font used for writing on the screen.

The font is used for both `draw_text()` and `print()`.

Parameters: **font** (`Font`) – The font to use.

Example: See example in `print()`.

`screen.load_image`(*source*)

Clears this image, then draws the `source` image centered in the screen.

Parameters: **source** (*Image or str*) – The source `Image`. If the argument is a string, then the `source` image is loaded from file.

Show/hide example ▲

Example: Show an image on the screen.

```
#!/usr/bin/env pybricks-micropython

from pybricks.hubs import EV3Brick
from pybricks.tools import wait
from pybricks.media.ev3dev import Image, ImageFile

# It takes some time to load images from the SD card, so it is best to load
# them once at the beginning of a program like this:
ev3_img = Image(ImageFile.EV3_ICON)


# Initialize the EV3
ev3 = EV3Brick()


# Show an image
ev3.screen.load_image(ev3_img)

# Wait some time to look at the image
wait(5000)
```

`screen.draw_image`(*x*, *y*, *source*, *transparent=None*)

Draws the `source` image on the screen.

Parameters:

- **x** (*int*) – The x-axis value where the left side of the image will start.
- **y** (*int*) – The y-axis value where the top of the image will start.
- **source** (*Image or str*) – The source `Image`. If the argument is a string, then the `source` image is loaded from file.
- **transparent** (*Color*) – The color of `image` to treat as transparent or `None` for no transparency.

`screen.draw_pixel`(*x*, *y*, *color=Color.BLACK*)

Draws a single pixel on the screen.

Parameters:

- **x** (*int*) – The x coordinate of the pixel.
- **y** (*int*) – The y coordinate of the pixel.
- **color** (*Color*) – The color of the pixel.

screen.draw_line(*x1*, *y1*, *x2*, *y2*, *width=1*, *color=Color.BLACK*)

Draws a line on the screen.

Parameters:

- **x1** (*int*) – The x coordinate of the starting point of the line.
- **y1** (*int*) – The y coordinate of the starting point of the line.
- **x2** (*int*) – The x coordinate of the ending point of the line.
- **y2** (*int*) – The y coordinate of the ending point of the line.
- **width** (*int*) – The width of the line in pixels.
- **color** (*Color*) – The color of the line.

Show/hide example ▲

Example: Draw some shapes on the screen.

```
#!/usr/bin/env pybricks-micropython

from pybricks.hubs import EV3Brick
from pybricks.tools import wait


# Initialize the EV3
ev3 = EV3Brick()


# Draw a rectangle
ev3.screen.draw_box(10, 10, 40, 40)

# Draw a solid rectangle
ev3.screen.draw_box(20, 20, 30, 30, fill=True)

# Draw a rectangle with rounded corners
ev3.screen.draw_box(50, 10, 80, 40, 5)

# Draw a circle
ev3.screen.draw_circle(25, 75, 20)

# Draw a triangle using lines
x1, y1 = 65, 55
x2, y2 = 50, 95
x3, y3 = 80, 95
ev3.screen.draw_line(x1, y1, x2, y2)
ev3.screen.draw_line(x2, y2, x3, y3)
ev3.screen.draw_line(x3, y3, x1, y1)

# Wait some time to look at the shapes
wait(5000)
```

screen.draw_box(*x1*, *y1*, *x2*, *y2*, *r=0*, *fill=False*, *color=Color.BLACK*)

Draws a box on the screen.

Parameters:

- **x1** (*int*) – The x coordinate of the left side of the box.
- **y1** (*int*) – The y coordinate of the top of the box.
- **x2** (*int*) – The x coordinate of the right side of the box.
- **y2** (*int*) – The y coordinate of the bottom of the box.
- **r** (*int*) – The radius of the corners of the box.
- **fill** (*bool*) – If `True`, the box will be filled with `color`, otherwise only the outline of the box will be drawn.
- **color** (*Color*) – The color of the box.

Example: See example in `draw_line()`.

`screen.draw_circle`(*x*, *y*, *r*, *fill=False*, *color=Color.BLACK*)

Draws a circle on the screen.

Parameters:

- **x** (*int*) – The x coordinate of the center of the circle.
- **y** (*int*) – The y coordinate of the center of the circle.
- **r** (*int*) – The radius of the circle.
- **fill** (*bool*) – If `True`, the circle will be filled with `color`, otherwise only the circumference will be drawn.
- **color** (*Color*) – The color of the circle.

Example: See example in `draw_line()`.

`screen.width`

Gets the width of the screen in pixels.

`screen.height`

Gets the height of the screen in pixels.

`screen.save`(*filename*)

Saves the screen as a `.png` file.

Parameters: **filename** (*str*) – The path to the file to be saved.

Raises:

- `TypeError` – `filename` is not a string.
- `OSError` – There was a problem saving the file.

Using the battery

`battery.voltage`()

Gets the voltage of the battery.

Returns: Battery voltage.

Return type: voltage: mV

`battery.current`()

Gets the current supplied by the battery.

Returns: Battery current.

Return type: current: mA