

Dokumentation zum LED-Display (20x*10y) von Tinkerforge-Blinkenlights programmiert mit Python 3

Autorin: Fabienne Calderara

Zugehörige Files:

Klassen

- LEDmatrix_init_171110.py
- LEDmatrix_moveableImage_171110.py
- LEDmatrix_split_merge_171110.py

Module

- imageProperties.py
- matrixProperties.py

Tests

- Test_init_171110.py
- Test_moveableImage_171110.py
- Test_split_merge_171110.py

Testergebnisse

- video_test_init.mp4
- video_test_moveableImage_v1.mp4
- video_test_moveableImage_v2.mp4
- video_test_moveableImage_v3.mp4
- video_test_split_merge.mp4
- video_test_split_merge_colorHeartrate_Variante.mp4
- video_test_split_merge_images.mp4
- video_test_split_merge_moves.mp4

Ordner

- tinkerforge
von tinkerforge_python_bindings_2_1_14
- __pycache__
- _Test_Log (mit den Test-Protokollen)
- Flask
 - bin
 - lib
 - static
 - templates
 - pip.selfcheck.json

Nummer der vorliegenden Fassung: V2.0	Verfasst von: Fabienne Calderara (caf1)	Genehmigung durch:
--	---	--------------------

Inhaltsverzeichnis

Aus dem Python-Tutorial.....	4
Konfiguration/Tools	4
Konzept.....	5
Eigenschaften des LED-Displays von Tinkerforge/Blinkenlights.....	6
LEDmatrix_init_171110.py: LEDmatrixConnect().....	7
Instanzvariablen	7
Objekte	8
Private Methoden.....	9
Public Methoden	11
#1 connect(self).....	11
#2 finish(self).....	11
#3 setFrameDuration(self, frequenz: int).....	11
#4 getFrameDuratin(self) → int.....	11
#5 light(self, status: bool).....	12
#6 setImage(self, image: list)	12
#7 setImageRGBindex(self, red: int, green: int, blue: int)	12
Testfile: Test_init_171110 Letztmals erfolgreich: 30.11.2017	13
Ergebnis Test LEDmatrixConnect().....	13
LEDmatrix_moveableImage_171110.py: MoveableImage()	14
Instanzvariablen	14
Objekte	15
Private Methoden.....	16
Public Methoden	18
#8 finish(self).....	18
#9 getMatrixPosition(self) → list	18
#10 getXvalues(self) → list.....	18
#11 getYvalues(self) → list.....	18
#39 getName(self) → str.....	19
#12 getImage(self) → list	19
#13 getLightOffImage(self) → list	19
#14 setLightOffImage(self, image: list)	20
#15 setLightOffImageColor(self, red: int, green: int, blue: int)	20
#16 resetLightOffImage(self)	20
#17 setImageFormat(self, imageFormat: 'portrait' 'landscape' = None)	21
#18 setInputImage(self, image: list, imageFormat: 'portrait' 'landscape' = None)	21
#19 getImageSizeAsText(self) → str	21
#20 getImageSize(self) → list.....	22
#21 startBlink(self)	22
#22 stopBlink(self).....	22
#23 startMove(self).....	22
#24 waiting(self).....	22
#25 stopMove(self)	23
#26 resetMove(self)	23
#27 moveRight(self)	23
#28 moveLeft(self)	23
#29 stopMoveHorizontal(self)	24
#30 moveUp(self).....	24
#31 moveDown(self)	24
#32 stop MoveVertical(self).....	24
#33 getMoveInfo(self) → list	25
#34 setMoveTime(self, second: float = 1.0)	25
#35 setRate(self, hertz: float = 1.0).....	26
#36 setPuls(self, puls: int)	26
#37 setTimingRatio(self, rationOn: int, ratioOff: int)	26
#38 setHeartrateFlag(self, boolean: bool = True)	27
#40 light(self, state: bool = True)	27
Testfile: Test_moveableImage_171110 Letztmals erfolgreich: 01.12..2017	28

LEDmatrix_split_merge_171110.py: LEDmatrix()	34
Objekte	34
Private Methoden.....	35
Public Methoden	37
#41 setBGcolor(self, red: int, green: int, blue: int).....	37
#42 getObjects(self) → list.....	37
#43 getMatrix(self) → LEDmatrixConnect	37
#45 getCase(self) → str	37
#44 finish(self).....	38
Darstellung Use Cases.....	38
Testfile: Test_split_merge_171110 Letztmals erfolgreich: 30.11.2017	39
Case 0: LEDmatrix()	39
Case 1: LEDmatrix('square')	40
Case 2: LEDmatrix('colorHeartrate')	41
Case 3: LEDmatrix('quarters4')	43
Case 4: LEDmatrix('row2').....	44
Case 5: LEDmatrix('col2')	45
Case 6: LEDmatrix('single').....	46
Testfile: Test_split_merge_171110_images Letztmals erfolgreich: 01.12.2017	47
Testfile: Test_split_merge_171110_moves Letztmals erfolgreich: 01.12.2017.....	48
Modul: imageProperties.....	49
imageProperties-Bilder	49
Modul: matrixProperties	50
Anhang A: Konsolen-Output der Testergebnisse von Test_split_merge_171110.py	52
Ergebnis Case 0: LEDmatrix()	52
Ergebnis Case 1: LEDmatrix('square')	52
Ergebnis Case 2: LEDmatrix('colorHeartrate).....	53
Ergebnis Case 3: LEDmatrix('quarters4).....	54
Ergebnis Case 4: LEDmatrix('row2)	55
Ergebnis Case 5: LEDmatrix('col2')	56
Ergebnis Case 6: LEDmatrix('single')	56
Anhang B: Informations-Suchergebnisse	57
Python	57
Flask	57
Json	57
REST-API	57
Tkinter.....	57
Tinkerforge - Blinkenlights	57
Bluetooth 4.0	58
Versionskontrolle	59

Aus dem Python-Tutorial

<https://www.python-kurs.eu/klassen.php> (abgerufen 28.11.2017)

Namen	Bezeichnung	Bedeutung
name	Public	Attribute ohne führende Unterstriche sind sowohl innerhalb einer Klasse als auch von außen les- und schreibbar.
_name	Protected	Man kann zwar auch von außen lesend und schreibend zugreifen, aber der Entwickler macht damit klar, dass man diese Member nicht benutzen sollte.
__name	Private	Sind von außen nicht sichtbar und nicht benutzbar.

Konstruktor

Genaugenommen gibt es in Python keine expliziten Konstruktoren oder Destruktoren. Häufig wird die `__init__`-Methode als Konstruktor bezeichnet. Wäre sie wirklich ein Konstruktor, würde sie wahrscheinlich `__constr__` oder `__constructor__` heißen. Sie heißt stattdessen `__init__`, weil mit dieser Methode ein Objekt, welcher vorher automatisch erzeugt ("konstruiert") worden ist, initialisiert wird. Diese Methode wird also unmittelbar nach der Konstruktion eines Objektes aufgerufen. Es wirkt also so, als würde das Objekt durch `__init__` erzeugt. Dies erklärt den häufig gemachten Fehler in der Bezeichnungsweise.

Wir benutzen nun die `__init__`-Methode, um die Objekte unserer Kontoklasse zu initialisieren. `__init__`-Konstruktoren werden wie andere Methoden definiert:

```
def __init__(self, inhaber, kontonummer,
             kontostand, kontokorrent=0):
    self.Inhaber = inhaber
    self.Kontonummer = kontonummer
    self.Kontostand = kontostand
    self.Kontokorrent = kontokorrent
```

Konfiguration/Tools

Tinkerforge, Blinkenlights

Brick Viewer (brickv): <https://www.tinkerforge.com/de/doc/Software/Brickv.html>

Brick Daemon (brickd): <https://www.tinkerforge.com/de/doc/Software/Brickd.html>

PyCharm CE: <https://www.jetbrains.com/pycharm/download/#section=mac>

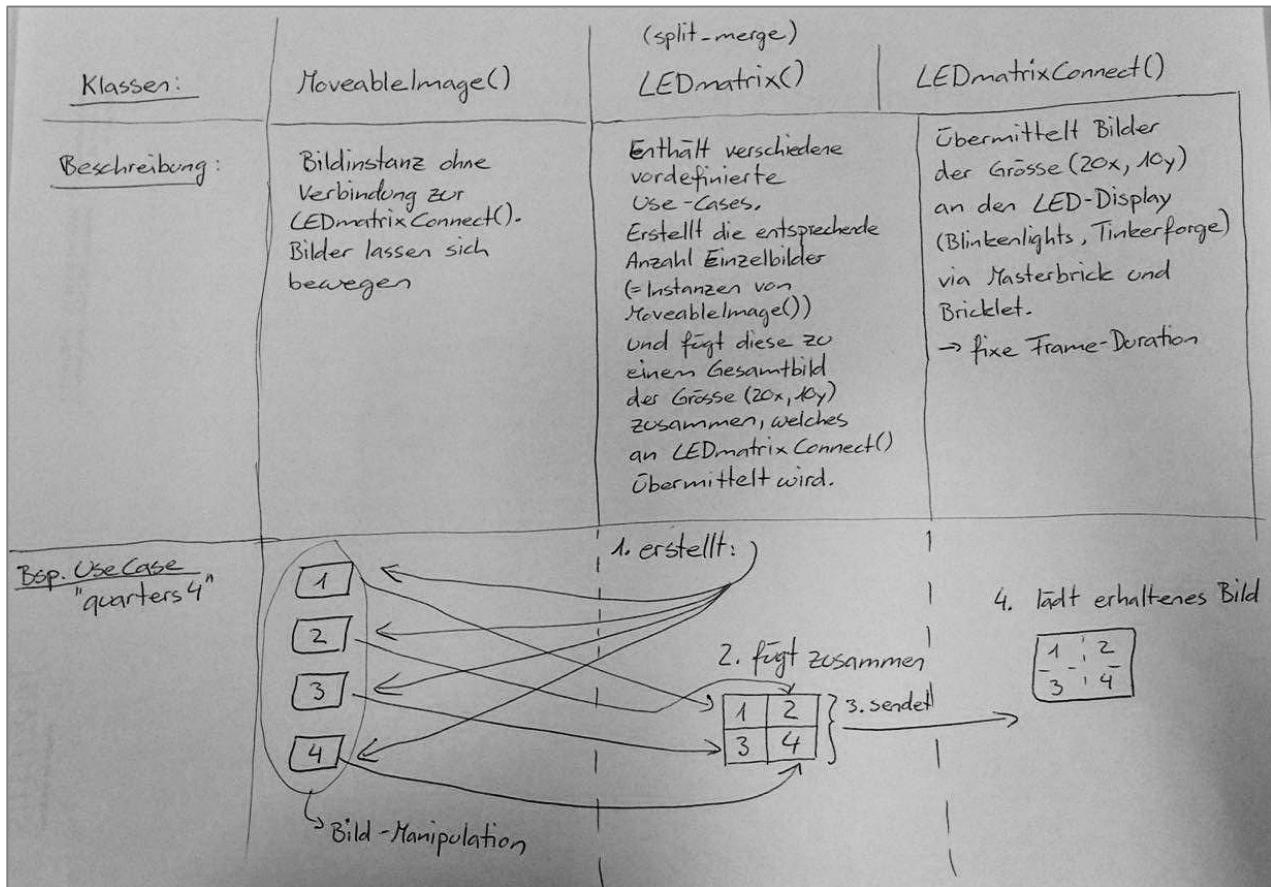
Syntax und Konventionen mit PyCharm CE überprüft. (Jedoch nicht alle Konventionen eingehalten.)

Python: IDLE

<https://docs.python.org/3/library/idle.html#>

Releases for Mac OS X: <https://www.python.org/downloads/mac-osx/>

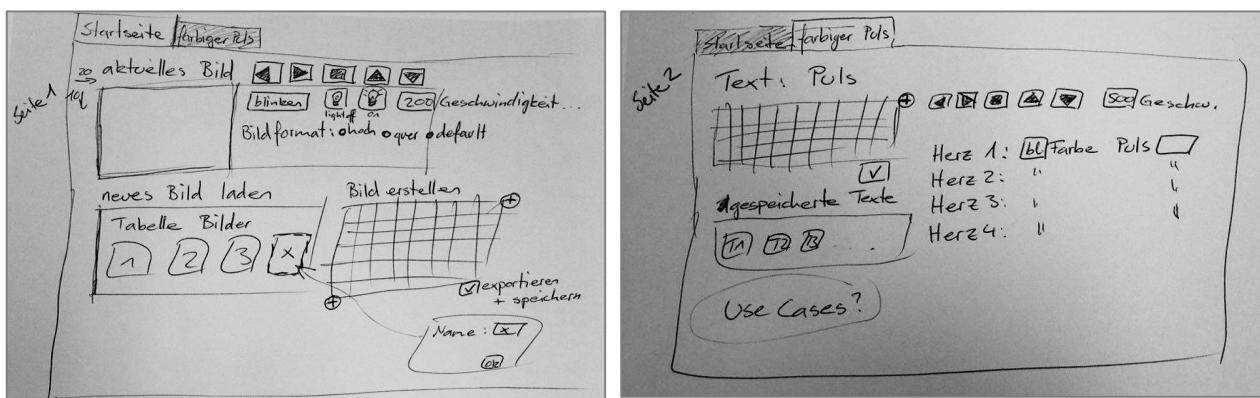
Konzept



Überlegung zur url für die REST-API:

- Für die meisten Methoden würde ich GET (etwas holen) und PUT (etwas verändern) verwenden.
- POST (create) würde ich nur für die Instanzierung der Klassen-Objekte verwenden.

GUI-Skizze:



Überlegungen zur Pulsübermittlung via Bluetooth 4.0

Mit LightBlue (<https://itunes.apple.com/ch/app/lightblue/id639944780?mt=12>, abgerufen 15.12.2017) können die Signale der Bluetooth-Uhren (MioAlpha) mit einem MacBook (z.B. MacBook Air) abgefangen werden. Die Herausforderung liegt darin, 4 Pulsuhrten gleichzeitig zu überwachen und immer den aktuellen Puls an das Programm zu übermitteln. (split-merge: Use Case 'colorHeartrate')

Eigenschaften des LED-Displays von Tinkerforge/Blinkenlights

LED-Nummerierung

190	189	170	169	150	149	130	129	110	109	90	89	70	69	50	49	30	29	10	9
191	188	171	168	151	148	131	128	111	108	91	88	71	68	51	48	31	28	11	8
192	187	172	167	152	147	132	127	112	107	92	87	72	67	52	47	32	27	12	7
193	186	173	166	153	146	133	126	113	106	93	86	73	66	53	46	33	26	13	6
194	185	174	165	154	145	134	125	114	105	94	85	74	65	54	45	34	25	14	5
195	184	175	164	155	144	135	124	115	104	95	84	75	64	55	44	35	24	15	4
196	183	176	163	156	143	136	123	116	103	96	83	76	63	56	43	36	23	16	3
197	182	177	162	157	142	137	122	117	102	97	82	77	62	57	42	37	22	17	2
198	181	178	161	158	141	138	121	118	101	98	81	78	61	58	41	38	21	18	1
199	180	179	160	159	140	139	120	119	100	99	80	79	60	59	40	39	20	19	0

Zuordnung [x/y]

[0/0]	[1/0]	[2/0]	[3/0]	[4/0]	[5/0]	[6/0]	[7/0]	[8/0]	[9/0]	[10/0]	[11/0]	[12/0]	[13/0]	[14/0]	[15/0]	[16/0]	[17/0]	[18/0]	[19/0]
[0/1]	[1/1]	[2/1]	[3/1]	[4/1]	[5/1]	[6/1]	[7/1]	[8/1]	[9/1]	[10/1]	[11/1]	[12/1]	[13/1]	[14/1]	[15/1]	[16/1]	[17/1]	[18/1]	[19/1]
[0/2]	[1/2]	[2/2]	[3/2]	[4/2]	[5/2]	[6/2]	[7/2]	[8/2]	[9/2]	[10/2]	[11/2]	[12/2]	[13/2]	[14/2]	[15/2]	[16/2]	[17/2]	[18/2]	[19/2]
[0/3]	[1/3]	[2/3]	[3/3]	[4/3]	[5/3]	[6/3]	[7/3]	[8/3]	[9/3]	[10/3]	[11/3]	[12/3]	[13/3]	[14/3]	[15/3]	[16/3]	[17/3]	[18/3]	[19/3]
[0/4]	[1/4]	[2/4]	[3/4]	[4/4]	[5/4]	[6/4]	[7/4]	[8/4]	[9/4]	[10/4]	[11/4]	[12/4]	[13/4]	[14/4]	[15/4]	[16/4]	[17/4]	[18/4]	[19/4]
[0/5]	[1/5]	[2/5]	[3/5]	[4/5]	[5/5]	[6/5]	[7/5]	[8/5]	[9/5]	[10/5]	[11/5]	[12/5]	[13/5]	[14/5]	[15/5]	[16/5]	[17/5]	[18/5]	[19/5]
[0/6]	[1/6]	[2/6]	[3/6]	[4/6]	[5/6]	[6/6]	[7/6]	[8/6]	[9/6]	[10/6]	[11/6]	[12/6]	[13/6]	[14/6]	[15/6]	[16/6]	[17/6]	[18/6]	[19/6]
[0/7]	[1/7]	[2/7]	[3/7]	[4/7]	[5/7]	[6/7]	[7/7]	[8/7]	[9/7]	[10/7]	[11/7]	[12/7]	[13/7]	[14/7]	[15/7]	[16/7]	[17/7]	[18/7]	[19/7]
[0/8]	[1/8]	[2/8]	[3/8]	[4/8]	[5/8]	[6/8]	[7/8]	[8/8]	[9/8]	[10/8]	[11/8]	[12/8]	[13/8]	[14/8]	[15/8]	[16/8]	[17/8]	[18/8]	[19/8]
[0/9]	[1/9]	[2/9]	[3/9]	[4/9]	[5/9]	[6/9]	[7/9]	[8/9]	[9/9]	[10/9]	[11/9]	[12/9]	[13/9]	[14/9]	[15/9]	[16/9]	[17/9]	[18/9]	[19/9]

Zuordnung [x/y] + LED-Nummerierung

[0/0]	[1/0]	[2/0]	[3/0]	[4/0]	[5/0]	[6/0]	[7/0]	[8/0]	[9/0]	[10/0]	[11/0]	[12/0]	[13/0]	[14/0]	[15/0]	[16/0]	[17/0]	[18/0]	[19/0]
190	189	170	169	150	149	130	129	110	109	90	89	70	69	50	49	30	29	10	9
[0/1]	[1/1]	[2/1]	[3/1]	[4/1]	[5/1]	[6/1]	[7/1]	[8/1]	[9/1]	[10/1]	[11/1]	[12/1]	[13/1]	[14/1]	[15/1]	[16/1]	[17/1]	[18/1]	[19/1]
191	188	171	168	151	148	131	128	111	108	91	88	71	68	51	48	31	28	11	8
[0/2]	[1/2]	[2/2]	[3/2]	[4/2]	[5/2]	[6/2]	[7/2]	[8/2]	[9/2]	[10/2]	[11/2]	[12/2]	[13/2]	[14/2]	[15/2]	[16/2]	[17/2]	[18/2]	[19/2]
192	187	172	167	152	147	132	127	112	107	92	87	72	67	52	47	32	27	12	7
[0/3]	[1/3]	[2/3]	[3/3]	[4/3]	[5/3]	[6/3]	[7/3]	[8/3]	[9/3]	[10/3]	[11/3]	[12/3]	[13/3]	[14/3]	[15/3]	[16/3]	[17/3]	[18/3]	[19/3]
193	186	173	166	153	146	133	126	113	106	93	86	73	66	53	46	33	26	13	6
[0/4]	[1/4]	[2/4]	[3/4]	[4/4]	[5/4]	[6/4]	[7/4]	[8/4]	[9/4]	[10/4]	[11/4]	[12/4]	[13/4]	[14/4]	[15/4]	[16/4]	[17/4]	[18/4]	[19/4]
194	185	174	165	154	145	134	125	114	105	94	85	74	65	54	45	34	25	14	5
[0/5]	[1/5]	[2/5]	[3/5]	[4/5]	[5/5]	[6/5]	[7/5]	[8/5]	[9/5]	[10/5]	[11/5]	[12/5]	[13/5]	[14/5]	[15/5]	[16/5]	[17/5]	[18/5]	[19/5]
195	184	175	164	155	144	135	124	115	104	95	84	75	64	55	44	35	24	15	4
[0/6]	[1/6]	[2/6]	[3/6]	[4/6]	[5/6]	[6/6]	[7/6]	[8/6]	[9/6]	[10/6]	[11/6]	[12/6]	[13/6]	[14/6]	[15/6]	[16/6]	[17/6]	[18/6]	[19/6]
196	183	176	163	156	143	136	123	116	103	96	83	76	63	56	43	36	23	16	3
[0/7]	[1/7]	[2/7]	[3/7]	[4/7]	[5/7]	[6/7]	[7/7]	[8/7]	[9/7]	[10/7]	[11/7]	[12/7]	[13/7]	[14/7]	[15/7]	[16/7]	[17/7]	[18/7]	[19/7]
197	182	177	162	157	142	137	122	117	102	97	82	77	62	57	42	37	22	17	2
[0/8]	[1/8]	[2/8]	[3/8]	[4/8]	[5/8]	[6/8]	[7/8]	[8/8]	[9/8]	[10/8]	[11/8]	[12/8]	[13/8]	[14/8]	[15/8]	[16/8]	[17/8]	[18/8]	[19/8]
198	181	178	161	158	141	138	121	118	101	98	81	78	61	58	41	38	21	18	1
[0/9]	[1/9]	[2/9]	[3/9]	[4/9]	[5/9]	[6/9]	[7/9]	[8/9]	[9/9]	[10/9]	[11/9]	[12/9]	[13/9]	[14/9]	[15/9]	[16/9]	[17/9]	[18/9]	[19/9]

Um Bilder zu generieren, kann im Moment folgender Link verwendet werden. (erstellt durch Jonathan Meier)
<http://ledmatrixeditor.i4mi.bfh.ch/>

LEDmatrix_init_171110.py: LEDmatrixConnect()

Klasse: LEDmatrixConnect(self)

REST-API: `@LEDmatrix.route('/ledmatrix.bfh.ch/api/v1.0/LEDmatrixConnect[POST])`

Verbindung zum LED-Display: **Test_init_171110.py zweimal laufen lassen!**

Imports

- from tinkerforge.ip_connection import IPConnection
- from tinkerforge.bricklet_led_strip import BrickletLEDStrip
- from tinkerforge.brick_master import BrickMaster
- import matrixProperties
- import time

Instanzvariablen

Attributname	Zuordnung	Beschreibung
<code>self.__UIDmaster</code> <i>Datentyp: str</i>	<code>matrixProperties.UIDmaster</code> „6et15y“	Für die Tinkerforge-Verbindung. Über brickv* ersichtlich.
<code>self.__UIDbricklet</code> <i>Datentyp: str</i>	<code>matrixProperties.UIDbricklet</code> „wVj“	Für die Tinkerforge-Verbindung. Über brickv* ersichtlich.
<code>self.__HOST</code> <i>Datentyp: undefined</i>	<code>matrixProperties.HOST</code> „localhost“	Für die Tinkerforge-Verbindung. Über brickv* ersichtlich.
<code>self.__PORT</code> <i>Datentyp: int</i>	<code>matrixProperties.PORT</code> 4223	Für die Tinkerforge-Verbindung. Über brickv* ersichtlich.
<code>self.__matrixIndexRed</code> <i>Datentyp: int</i>	<code>matrixProperties.IndexRed</code> 0	Die Farben werden im Array wie folgt gespeichert: [red, green, blue] Beispiel: gelb = [255, 255, 0]
<code>self.__matrixIndexGreen</code> <i>Datentyp: int</i>	<code>matrixProperties.IndexGreen</code> 1	Die Farben werden im Array wie folgt gespeichert: [red, green, blue] Beispiel: gelb = [255, 255, 0]
<code>self.__matrixIndexBlue</code> <i>Datentyp: int</i>	<code>matrixProperties.IndexBlue</code> 2	Die Farben werden im Array wie folgt gespeichert: [red, green, blue] Beispiel: gelb = [255, 255, 0]
<code>self.__rows</code> <i>Datentyp: int</i>	<code>matrixProperties.ROWS</code> 10	Beschreibt die Anzahl Zeilen. (y-Achse)
<code>self.__columns</code> <i>Datentyp: int</i>	<code>matrixProperties.COLUMNS</code> 20	Beschreibt die Anzahl Spalten. (x-Achse)
<code>self.__LEDnr</code> <i>Datentyp: str</i>	<code>matrixProperties.LEDnr</code> [[190, 191, 192, 193, 194, 195, 196, 197, 198, 199], [189, 188, 187, 186, 185, 184, 183, 182, 181, 180], [170, 171, 172, 173, 174, 175, 176, 177, 178, 179], [169, 168, 167, 166, 165, 164, 163, 162, 161, 160], [150, 151, 152, 153, 154, 155, 156, 157, 158, 159], [149, 148, 147, 146, 145, 144, 143, 142, 141, 140], [130, 131, 132, 133, 134, 135, 136, 137, 138, 139], [129, 128, 127, 126, 125, 124, 123, 122, 121, 120], [110, 111, 112, 113, 114, 115, 116, 117, 118, 119], [109, 108, 107, 106, 105, 104, 103, 102, 101, 100], [90, 91, 92, 93, 94, 95, 96, 97, 98, 99], [89, 88, 87, 86, 85, 84, 83, 82, 81, 80], [70, 71, 72, 73, 74, 75, 76, 77, 78, 79], [69, 68, 67, 66, 65, 64, 63, 62, 61, 60], [50, 51, 52, 53, 54, 55, 56, 57, 58, 59], [49, 48, 47, 46, 45, 44, 43, 42, 41, 40], [30, 31, 32, 33, 34, 35, 36, 37, 38, 39], [29, 28, 27, 26, 25, 24, 23, 22, 21, 20], [10, 11, 12, 13, 14, 15, 16, 17, 18, 19], [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]]	<p>Die LED-Nummern wird für folgenden Funktionsaufruf benötigt:</p> <pre>self._brickletLEDstrip.set_rgb_values (self.__LEDnr[xMatrix][y], self.__rows, self.__rgb[self.__matrixIndexRed], self.__rgb[self.__matrixIndexGreen], self.__rgb[self.__matrixIndexBlue])</pre> <p>→ set_rgb_values(index, length, r, g, b)</p> <ul style="list-style-type: none"> • index = 1. Lämpchen, welches angesteuert wird • length = Anzahl Lämpchen, welche verändert werden (Hier wird die Zahl der Zeilen genommen) • r, g, b: verlangen 16er-Arrays, daher gibt es <code>self.__rgb</code> und <code>self.__numLEDS</code>

Attributname	Zuordnung	Beschreibung
self.__numLEDS Datentyp: int	matrixProperties.NUM_LEDS 16	Für die Erstellung des Arrays self.__rgb, weil die r, g, b in der set_rgb_values(index, length, r, g, b) – Methode 16er-Arrays verlangen.
self.__rgb Datentyp: list	[[0 for i in range(self.__numLEDS)] for i in range (3)] = [[0 for i in range(16)] for i in range (3)]	Weil die r, g, b in der set_rgb_values(index, length, r, g, b) – Methode 16er-Arrays verlangen.
self.__image Datentyp: list	matrixProperties.defaultImage Format: [[[red, green, blue] for i in range(x columns)] for i in range(y rows)] Beispiel: [[[int, int, int] for i in range(20)] for i in range(10)]	Standardbild: „HI !“ Der Array des Bildes muss die Farbinformationen pro LED-Lämpchen enthalten. Das Format des Arrays ist der Spalte „Zuordnung“ zu entnehmen. Die RGB-Farbwerte (red, green, blue) müssen ein Integer sein und im Wertebereich von 0 bis 255 liegen.
self.__imageIndexRed Datentyp: int	0	Die Farben werden im Array wie folgt gespeichert: [red, green, blue] Beispiel: gelb = [255, 255, 0] <i>Separat aufgeführt, falls Bilder übermittelt werden, deren Farbinformationen nicht im Format [red, green, blue] gespeichert sind.</i>
self.__imageIndexGreen Datentyp: int	1	Die Farben werden im Array wie folgt gespeichert: [red, green, blue] Beispiel: gelb = [255, 255, 0] <i>Separat aufgeführt, falls Bilder übermittelt werden, deren Farbinformationen nicht im Format [red, green, blue] gespeichert sind.</i>
self.__imageIndexBlue Datentyp: int	2	Die Farben werden im Array wie folgt gespeichert: [red, green, blue] Beispiel: gelb = [255, 255, 0] <i>Separat aufgeführt, falls Bilder übermittelt werden, deren Farbinformationen nicht im Format [red, green, blue] gespeichert sind.</i>
self.__light Datentyp: bool	True	Varianten: True / False True: Licht an (on) False: Licht aus (off)

* <https://www.tinkerforge.com/de/doc/Software/Brickv.html> (abgerufen 28.11.2017)

Objekte

Attributname	Zuordnung	Beschreibung
self.__ipcon Datentyp: IPConnection	IPConnection()	Kreiert eine IP-Verbindung. from tinkerforge.ip_connection import IPConnection
self.__masterBrick Datentyp: BrickMaster	BrickMaster(self.__UIDmaster, self.__ipcon)	from tinkerforge.brick_master import BrickMaster
self.__brickletLEDstrip Datentyp: BrickletLEDStrip	BrickletLEDStrip(self.__UIDbricklet, self.__ipcon)	from tinkerforge.bricklet_led_strip import BrickletLEDStrip

Private Methoden

__init__(self) => Konstruktor

Enthält die oben aufgeführten Instanzvariablen und Objekte. Diese Methode dient der Initialisierung.

Aufruf	Beschreibung
self.__ipcon.connect(self.__HOST, self.__PORT)	Verbindung herstellen
self.__brickletLEDstrip.set_frame_duration(20)	20ms ergibt 20 Bilder pro Sekunde
self.__brickletLEDstrip.set_channel_mapping(self.__brickletLEDstrip.CHANNEL_MAPPING_RGB)	CHANNEL_MAPPING_RGB = 6 Farbkanal definieren (RGB statt BGR)

```
self.__brickletLEDstrip.register_callback(self.__brickletLEDstrip.CALLBACK_FRAME_RENDERED,  
                                         lambda x: self.__loadPicture__())
```

Beschreibung: (Ich kenne mich mit lambda nicht wirklich aus, aber es funktioniert)

Der obere Methodenaufruf mit der lambda-Expression generiert einen Callback der die Methode __loadPicture__(self)

aufruft, mit einer definierten Frame-Duration (self.__brickletLEDstrip.set_frame_duration(self.__frameDuration))

Funktion: Das self.__image wird bei jedem Callback „geladen“ / an den LED-Display übermittelt.

__loadPicture__(self)

```
94     def __loadPicture__(self):  
95         """ Methode lädt ein ganzes Bild, sofern self.__light auf True gesetzt ist => Licht an. """  
96  
97         if (self.__light is True):  
98             xMatrix = 0  
99             yMatrix = 0  
100            xImage = 0  
101            columns = self.__columns # 20  
102            while (columns > 0):  
103                if (xMatrix % 2 == 0): # Farbwerte von image-array nach rgb-array bei gerader Spalte  
104                    y = 0  
105                    self.__fillAsc__(xImage, yMatrix)  
106                else: # Farbwerte von image-array nach rgb-array bei ungerader Spalte  
107                    y = self.__rows - 1  
108                    self.__fillDesc__(xImage, self.__rows - (yMatrix + 1))  
109  
110            # Aufrufen der Methode set_rgb_values(index, length, r, g, b). (r, g, b, verlangen 16er-Array's)  
111            self.__brickletLEDstrip.set_rgb_values(self.__LEDnr[xMatrix][y], self.__rows,  
112                                            self.__rgb[self.__matrixIndexRed],  
113                                            self.__rgb[self.__matrixIndexGreen],  
114                                            self.__rgb[self.__matrixIndexBlue])  
115  
116            columns -= 1  
117            xImage += 1  
118            xMatrix += 1  
119            if xMatrix >= self.__columns:  
120                xMatrix = 0  
121                xImage = 0
```

self.__row enthält den Wert 10

- Diese Funktion wird vom *Callback* der __init__() aufgerufen.
- Funktion macht nur etwas, wenn self.__light auf True gesetzt ist.
- Das Bild wird von links [0/0] nach rechts [19/9] in den Display geladen (siehe unteres Bild).
- Die Farbwerte werden pro Kolonne (Spalte) abgefüllt, bei gerader Kolonnen-Nummer aufsteigend (Zeile 104-106), bei ungerader Kolonnen-Nummer absteigend (Zeile 107-109).
- Von Zeile 104 bis 109 werden die Farbinformationen des Bildes in den Array (self.__rgb) abgefüllt. Von Zeile 112 bis 115 werden die Farbinformationen des self.__rgb den Farbkanälen entsprechend übergeben.
- Diese Funktion ruft folgende privaten Methoden auf:
 - self.__fillDesc__(self, xImage, yMatrix)
=> Beispiel: self.__fillDesc__(0, 0) <= für (xImage, yMatrix)
 - self.__fillAsc__(self, xImage, yMatrix)
=> Beispiel: self.__fillAsc__(1, 9) <= für (xImage, self.__rows - (yMatrix + 1)) <= (1, 10 - (0 + 1))

Zuordnung [x/y] + LED-Nummerierung

[0/0]	[1/0]	[2/0]	[3/0]	[4/0]	[5/0]	[6/0]	[7/0]	[8/0]	[9/0]	[10/0]	[11/0]	[12/0]	[13/0]	[14/0]	[15/0]	[16/0]	[17/0]	[18/0]	[19/0]
190	189	170	169	150	149	130	129	110	109	90	89	70	69	50	49	30	29	10	9
[0/1]	[1/1]	[2/1]	[3/1]	[4/1]	[5/1]	[6/1]	[7/1]	[8/1]	[9/1]	[10/1]	[11/1]	[12/1]	[13/1]	[14/1]	[15/1]	[16/1]	[17/1]	[18/1]	[19/1]
191	188	171	168	151	148	131	128	111	108	91	88	71	68	51	48	31	28	11	8
[0/2]	[1/2]	[2/2]	[3/2]	[4/2]	[5/2]	[6/2]	[7/2]	[8/2]	[9/2]	[10/2]	[11/2]	[12/2]	[13/2]	[14/2]	[15/2]	[16/2]	[17/2]	[18/2]	[19/2]
192	187	172	167	152	147	132	127	112	107	92	87	72	67	52	47	32	27	12	7
[0/3]	[1/3]	[2/3]	[3/3]	[4/3]	[5/3]	[6/3]	[7/3]	[8/3]	[9/3]	[10/3]	[11/3]	[12/3]	[13/3]	[14/3]	[15/3]	[16/3]	[17/3]	[18/3]	[19/3]
193	186	173	166	153	146	133	126	113	106	93	86	73	66	53	46	33	26	13	6
[0/4]	[1/4]	[2/4]	[3/4]	[4/4]	[5/4]	[6/4]	[7/4]	[8/4]	[9/4]	[10/4]	[11/4]	[12/4]	[13/4]	[14/4]	[15/4]	[16/4]	[17/4]	[18/4]	[19/4]
194	185	174	165	154	145	134	125	114	105	94	85	74	65	54	45	34	25	14	5
[0/5]	[1/5]	[2/5]	[3/5]	[4/5]	[5/5]	[6/5]	[7/5]	[8/5]	[9/5]	[10/5]	[11/5]	[12/5]	[13/5]	[14/5]	[15/5]	[16/5]	[17/5]	[18/5]	[19/5]
195	184	175	164	155	144	135	124	115	104	95	84	75	64	55	44	35	24	15	4
[0/6]	[1/6]	[2/6]	[3/6]	[4/6]	[5/6]	[6/6]	[7/6]	[8/6]	[9/6]	[10/6]	[11/6]	[12/6]	[13/6]	[14/6]	[15/6]	[16/6]	[17/6]	[18/6]	[19/6]
196	183	176	163	156	143	136	123	116	103	96	83	76	63	56	43	36	23	16	3
[0/7]	[1/7]	[2/7]	[3/7]	[4/7]	[5/7]	[6/7]	[7/7]	[8/7]	[9/7]	[10/7]	[11/7]	[12/7]	[13/7]	[14/7]	[15/7]	[16/7]	[17/7]	[18/7]	[19/7]
197	182	177	162	157	142	137	122	117	102	97	82	77	62	57	42	37	22	17	2
[0/8]	[1/8]	[2/8]	[3/8]	[4/8]	[5/8]	[6/8]	[7/8]	[8/8]	[9/8]	[10/8]	[11/8]	[12/8]	[13/8]	[14/8]	[15/8]	[16/8]	[17/8]	[18/8]	[19/8]
198	181	178	161	158	141	138	121	118	101	98	81	78	61	58	41	38	21	18	1
[0/9]	[1/9]	[2/9]	[3/9]	[4/9]	[5/9]	[6/9]	[7/9]	[8/9]	[9/9]	[10/9]	[11/9]	[12/9]	[13/9]	[14/9]	[15/9]	[16/9]	[17/9]	[18/9]	[19/9]
199	180	179	160	159	140	139	120	119	100	99	80	79	60	59	40	39	20	19	0

fillDesc__(self, xImage, yMatrix)

```

82     def __fillDesc__(self, xImage, yMatrix):
83         """ Private Methode: Füllt eine Spalte für die LED-Matrix aus (von oben nach unten) """
84         yImage = 0
85         i = 0
86         while (i < self.__rows): # i < 10
87             self.__fillRBGs__(yMatrix, xImage, yImage)
88             i += 1
89             yImage += 1
90             yMatrix -= 1
91             if (yMatrix < 0):
92                 yMatrix = self.__rows - 1

```

- Diese Funktion wird von __loadPicture() aufgerufen.
- Diese Funktion ruft self.__fillRBGs__(self, yMatrix, xImage, yImage) auf.

fillAsc__(self, xImage, yMatrix)

```

70     def __fillAsc__(self, xImage, yMatrix):
71         """ Private Methode: Füllt eine Spalte für die LED-Matrix aus (von unten nach oben) """
72         yImage = 0
73         i = 0
74         while (i < self.__rows): # i < 10
75             self.__fillRBGs__(yMatrix, xImage, yImage)
76             i += 1
77             yImage += 1
78             yMatrix += 1
79             if (yMatrix >= self.__rows):
80                 yMatrix = 0

```

- Diese Funktion wird von __loadPicture() aufgerufen.
- Diese Funktion ruft self.__fillRBGs__(self, yMatrix, xImage, yImage) auf.

fillRBGs__(self, yMatrix, xImage, yImage)

```

64     def __fillRBGs__(self, yMatrix, xImage, yImage):
65         """ Private Methode: Füllt die Farbinformationen für einen Pixel in die einzelnen RGB-Arrays ab. """
66         self.__rgb[self.__matrixIndexRed][yMatrix] = self.__image[xImage][yImage][self.__imageIndexRed]
67         self.__rgb[self.__matrixIndexGreen][yMatrix] = self.__image[xImage][yImage][self.__imageIndexGreen]
68         self.__rgb[self.__matrixIndexBlue][yMatrix] = self.__image[xImage][yImage][self.__imageIndexBlue]

```

- Diese Funktion wird von self.__fillDesc__() oder von self.__fillAsc__() aufgerufen.

Public Methoden

#1 connect(self)

```
128     def connect(self):
129         """ Verbindungsauftbau """
130         self._ipcon.connect(self._HOST, self._PORT)
131         print("Verbindung zu brickd hergestellt.")
```

- Für die (Wieder-)Verbindung mit dem Blinkenlights-LED-Display.
- Die Funktion `connect()` stammt von `tinkerforge.ip_connection`.
- Im Testfile: # 1
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/LEDmatrixConnect/connect [PUT]`

#2 finish(self)

```
134     def finish(self):
135         """ Beendet die Verbindung. """
136         # input("Press key to exit\n")
137         self._ipcon.disconnect()
138         print("Verbindung beendet.")
```

- Für den Verbindungs-Abbau. Um das Programm danach zu beenden.
Bei Bedarf kann die `input`-Methode verwendet werden, damit der Verbindungsabbau erst mit der Tastatureingabe „Enter“ erfolgt. → `input("Press key to exit\n")`
- Die Funktion `disconnect()` stammt von `tinkerforge.ip_connection`.
- Im Testfile: # 2
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/LEDmatrixConnect/finish [PUT]`

#3 setFrameDuration(self, frequenz: int)

```
140     def setFrameDuration(self, millisec: int = 20):
141         """ Set frame duration to 50ms (20 frames per second).
142             Verändert die Anzeigedauer eines Frames.
143             :param millisec: int
144             """
145             self.__brickletLEDstrip.set_frame_duration(millisec)
```

- Diese Methode ändert die Anzeigedauer (in Millisekunden) eines Bildes (Frame), dadurch ergibt sich eine Frequenz von x Bildern pro Sekunde. Defaultwert = 20 ms.
- Beispiel: Bei 20 ms (default-Wert im aktuellen Programm), werden 50 Bilder pro Sekunde angezeigt.
- Die Funktion `set_frame_duration(ms)` stammt von `tinkerforge.bricklet_led_strip`.
- Im Testfile: # 3
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/LEDmatrixConnect/setVelocity/<int:frequenz> [PUT]`

#4 getFrameDuration(self) → int

```
147     def getFrameDuration(self) -> int:
148         """ Gibt einen int-Wert zurück. Anzeigedauer der Frames auf dem LED-Display.
149             (20ms entspricht 50 Bilder/Sekunde)
150             :return frame duration: int
151             """
152             return self.__brickletLEDstrip.get_frame_duration()
```

- Diese Methode gibt die aktuelle Frame-Duration zurück.
- Die Funktion `get_frame_duration()` stammt von `tinkerforge.bricklet_led_strip`.
- Im Testfile: # 4
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/LEDmatrixConnect/getFrameDuration [GET]`

#5 light(self, status: bool)

```

156     def light(self, status: bool = True): # status = boolean True / False
157         """ Licht-Regelung auf 'unterster' Stufe, wirkt sich auf den gesamten LED-Display.
158         True = 'Licht an', False = 'Licht aus'
159         :param status: boolean
160         """
161         self.__light = status
162
163         time.sleep(1)
164
165         if (status is False):
166             i = 0
167             while i < self.__columns: # hier 20
168                 self.__brickletLEDstrip.set_rgb_values(i * 10, 10, [0] * self.__numLEDS, [0] * self.__numLEDS,
169                                                 [0] * self.__numLEDS)
170                 i += 1

```

- Diese Methode "schaltet" das Licht ein und aus. Bei *True* ist das Licht an und die Bilder werden angezeigt. Wird das Licht auf *False* (Licht aus) gestellt, so wird einmalig ein leeres Bild (Farbwerte [0,0,0]) an die Matrix übermittelt. Bei *False* lädt die *__loadPicture__()* kein Bild mehr. (Mit einem *time.sleep()* scheint es etwas besser zu funktionieren. Ev. weil dann die *__loadPicture__()* nichts mehr lädt.)
- Defaultwert: *True* (Zu beachten (Hypothese): Wert sollte grösser sein als die *self._sleepTime* der *LEDmatrix*-Klasse von *LEDmatrix_split_merge_171110.py*)
- Im Testfile: # 5
- REST-API: @*LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/LEDmatrixConnect/light/<boolean>:status* [PUT])

#6 setImage(self, image: list)

```

170     def setImage(self, image: list):
171         """ Liste mit den RGB-Werten der
172             Form [[[red, green, blue] for i in range(x columns)] for i in range(y rows)]
173         :param image: list
174         """
175         self.__image = image

```

- Diese Methode ändert das Bild, welches mit *__loadPicture__()* an den LED-Display übermittelt wird. Der Array des Bildes muss die Farbinformationen pro LED-Lämpchen enthalten und benötigt folgendes Format: [[[red, green, blue] for i in range(x columns)] for i in range(y rows)] In unserem Beispiel: [[[int, int, int] for i in range(20)] for i in range(10)] Die RGB-Farbwerte (**red, green, blue**) müssen ein Integer sein und im Wertebereich **von 0 bis 255** liegen.
- Im Testfile: # 6
- REST-API: @*LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/LEDmatrixConnect/setImage/<list>:image* [PUT])

#7 setImageRGBIndex(self, red: int, green: int, blue: int)

```

178     def setImageRGBIndex(self, red: int, green: int, blue: int):
179         """ Werte: 0 bis 2, Default: red = 0, green = 1, blue = 2
180         :param red: int
181         :param green: int
182         :param blue: int
183         """
184         self.__imageIndexRed = red
185         self.__imageIndexGreen = green
186         self.__imageIndexBlue = blue

```

- Mit dieser Methode kann der Index für die RGB-Werte des Bildes geändert werden. Für RGB-Kanal ist es (0,1,2) für BGR entsprechend (2,1,0). Die Effekte sind im Testprogramm (*Test_init_171110*) ersichtlich.
- Die int-Werte müssen im Bereich von 0 bis 2 liegen.
- Im Testfile: # 7
- REST-API: @*LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/LEDmatrixConnect/setImageRGBIndex/<int>:red/<int>:green/<int>:blue* [PUT])

Testfile: Test_init_171110 Letztmals erfolgreich: 30.11.2017

Das folgende Testfile testet die Funktion des *LEDmatrix_init_171110.py* und explizit die 7 public-Methoden der Klasse *LEDmatrixConnect*.

```
1  from LEDmatrix_init_171110 import *
2  import imageProperties
3  import time
4
5
6  ▶ if __name__ == "__main__":
7      matrix = LEDmatrixConnect()
8      print(matrix.getFrameDuration())          # 4
9      matrix.setFrameDuration(50)               # 3
10     print(matrix.getFrameDuration())          # 4
11     time.sleep(3)
12     matrix.setImage(imageProperties.hearts1) # 6
13     time.sleep(1.5)
14     matrix.setImageRGBIndex(2, 2, 2)         # 7
15     time.sleep(2)
16     matrix.setImageRGBIndex(1, 1, 1)         # 7
17     time.sleep(2)
18     matrix.setImageRGBIndex(0, 0, 0)         # 7
19     time.sleep(2)
20     matrix.setImageRGBIndex(0, 1, 2)         # 7
21     time.sleep(2)
22
23     matrix.light(False)                     # 5
24     time.sleep(1.5)
25     matrix.light(True)                     # 5
26     time.sleep(1.5)
27     matrix.finish()                       # 2
28     time.sleep(1.5)
29     matrix.connect()                      # 1
30     matrix.setImage(imageProperties.forms1) # 6
31     time.sleep(1.5)
32     matrix.light(False)                   # 5
33     time.sleep(1)
34     matrix.finish()                      # 2
```

Ergebnis Test LEDmatrixConnect()

- ✓ Instanz LEDmatrix wurde erstellt.
- ✓ Verbindung zu brickd hergestellt.
- ✓ Callback aktiviert.

- ✓ 20
- ✓ 50

- ✓ Verbindung beendet.

- ✓ Verbindung zu brickd hergestellt.

- ✓ Verbindung beendet.

LEDmatrix_moveableImage_171110.py: MoveableImage()

Klasse: MoveableImage(self, name: str, mylImage: list=imageProperties.forms1, matrix_xValues: list=[0, 20], matrix_yValues: list=[0, 10])

REST-API:

```
@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/<str:name>/<list:mylImage>/<list:matrix_xValues>/<list:matrix_yValues>[POST]')
```

Imports:

- import imageProperties
- import threading
- import time

Instanzvariablen

Attributname	Zuordnung	Beschreibung
self.__objectName <i>Datentyp: str</i>	name	Benennung der Bildinstanz, damit Mensch diese wieder identifizieren kann.
self.__matrix_xValues <i>Datentyp: list</i>	matrix_xValues default: [0, 20]	Maximallänge = Displaylänge (Hier: 20)
self.__matrix_yValues <i>Datentyp: list</i>	matrix_yValues default: [0, 10]	Maximalhöhe = Displayhöhe (Hier: 10)
self.__inputlImage <i>Datentyp: list</i>	mylImage default: imageProperties.forms1	Das Default-Bild sollte idealerweise der Grösse der Bildinstanz entsprechen.
self.__outputlImage <i>Datentyp: list</i>	<code>[[[] for i in range(self.__matrix_yValues[1])] for i in range(self.__matrix_xValues[1])]</code> default: <code>[[[] for i in range(10)] for i in range(20)]</code>	Zuerst die Anzahl Zeilen angeben (y), dann die Anzahl Spalten (x). Default ergibt eine Matrix von 20x mal 10y
self.__lightOfflImage <i>Datentyp: list</i>	<code>[[[0 for i in range(3)] for i in range(self.__matrix_yValues[1])] for i in range(self.__matrix_xValues[1])]</code> default: <code>[[[0 for i in range(3)] for i in range(10)] for i in range(20)]</code>	Defaultmässig ein Bild mit den RGB-Farbinformationen (0,0,0) und der Grösse der Bildinstanz
self.__imageFormat <i>Datentyp: str</i>	None	Varianten: None / 'portrait' / 'landscape' Portrait = Hochformat Landscape = Querformat
self.__startX <i>Datentyp: int</i>	0	Benötigt in den privaten Methoden: <code>__loadImage__()</code> <code>__moveImage__()</code>
self.__startY <i>Datentyp: int</i>	0	Benötigt in den privaten Methoden: <code>__loadImage__()</code> <code>__moveImage__()</code>

Attributname	Zuordnung	Beschreibung
self.__move <i>Datentyp: bool</i>	False	Varianten: False / True
self.__moveHorizontal <i>Datentyp: str</i>	None	Varianten: None / 'left' / 'right'
self.__moveVertical <i>Datentyp: str</i>	None	Varianten: None / 'up' / 'down'
self.__moveTime <i>Datentyp: float</i>	1.0	Angabe in Sekunden. Bild wird alle x Sekunden bewegt. Umsetzung im Code: Attribut im time.sleep() in der __moveImage__()
self.__blinken <i>Datentyp: bool</i>	False	Varianten: False / True
self.__Hz <i>Datentyp: float</i>	1.0	Frequenz = Ereignisse pro Sekunde
self.__timingRatio <i>Datentyp: list</i>	[1, 1]	Taktverhältnis von Licht an : Licht aus
self.__on_off <i>Datentyp: list</i>	[0.5, 0.5]	Zeit von [Licht an, Licht aus] in Sekunden
self.__light <i>Datentyp: bool</i>	True	Varianten: False / True (Light on / off)
self.__heartrate <i>Datentyp: bool</i>	False	Varianten: False / True Flag für: „Die Frequenz stammt vom Puls.“
self.__heartrateOff <i>Datentyp: float</i> (Sekunden) PROTECTED (es kann von aussen zugegriffen werden)	0.5	Allenfalls auf 0.2 stellen. Fixer Wert für die „Licht aus“-Zeit beim blinken in der Pulsfrequenz.
self.__stopped <i>Datentyp: bool</i>	False	Varianten: False / True Wird von __blink__() und __moveImage__() benötigt.
self.__strMove <i>Datentyp: str</i>	"Thread move " + self.__objectName + " stopped.\n"	String für die Mitteilung, dass der Move-Thread beendet wurde. Verwendet von __moveImage__().
self.__strBlink <i>Datentyp: str</i>	"Thread blink " + self.__objectName + " stopped.\n"	String für die Mitteilung, dass der Blink-Thread beendet wurde. Verwendet von __blink__().

Objekte

Attributname	Zuordnung	Beschreibung
self.__threadBlink	threading.Thread(target=self.__blink__, args=())	Um das Bild blinken zu lassen.
self.__threadMove	threading.Thread(target=self.__moveImage__, args=())	Um das Bild zu bewegen.

Private Methoden

__init__(self) => Konstruktor

Enthält die oben aufgeführten Instanzvariablen und Objekte. Diese Methode dient der Initialisierung.

Aufruf	Beschreibung
<code>__loadImage()</code>	
<code>self.__threadBlink.daemon</code>	True
<code>self.__threadMove.daemon</code>	True
<code>self.__threadBlink.start()</code>	
<code>self.__threadMove.start()</code>	

__loadImage__(self)

```

62     def __loadImage__(self):
63         """ Lädt das outputImage einmalig bei der Initialisierung. """
64         x = 0
65         xMove = 0
66         yMove = 0
67         while x < len(self.__outputImage):
68             y = 0
69             while y < len(self.__outputImage[x]):
70                 self.__outputImage[x][y] = self.__inputImage[xMove][yMove]
71                 y += 1
72                 yMove = (yMove + 1) % len(self.__inputImage[xMove])
73             x += 1
74             xMove = (xMove + 1) % len(self.__inputImage)
75             yMove = 0
    
```

- Lädt das Input-Bild einmalig ins Output-Bild, wird bei der Initialisierung aufgerufen.
- Wird von `setInputImage(self, image, imageFormat)` aufgerufen.

__blink__(self)

```

122     def __blink__(self):
123         """ Methode in separatem Thread, welche für das Blinken zuständig ist.
124             Blinken erfolgt durch das 'ein und ausschalten' des Lichts.
125         """
126         while True:
127             if (self.__stopped is True):
128                 print(self.__strBlink)
129                 break
130
131             if (self.__blitzen is True):
132                 self.__light = True
133                 time.sleep(self.__on_off[0]) # für 'on'
134                 self.__light = False
135                 time.sleep(self.__on_off[1]) # für 'off'
    
```

- Lässt das Bild blinken, indem der Licht-Flag (`self.__light`) auf `True` oder `False` gesetzt wird, entsprechend den Angaben der `self.__on_off`-Variablen.

_moveImage__(self)

```

79      def __moveImage__(self):
80          """ Methode in separatem Thread, welche für die Bildbewegung zuständig ist.
81              inputImage bleibt unverändert, das outputImage wird bei jedem Durchgang angepasst.
82              Mögliche Bewegungen: Oben, unten, links, rechts und Kombinationen davon.
83          """
84          while True:
85              if (self.__stopped is True):
86                  print(self.__strMove)
87                  break
88
89              if (self.__move is True):
90                  xMove = 0
91                  yMove = 0
92                  if (self.__moveHorizontal is not None and self.__imageFormat != 'portrait'):
93                      if (self.__moveHorizontal == 'right'):
94                          xMove = (self.__startX + 1) % len(self.__inputImage)
95                      if (self.__moveHorizontal == 'left'):
96                          xMove = (self.__startX - 1) % len(self.__inputImage)
97                      self.__startX = xMove
98
99                  if (self.__moveVertical is not None and self.__imageFormat != 'landscape'):
100                     if (self.__moveVertical == 'up'):
101                         yMove = (self.__startY + 1) % len(self.__inputImage[xMove])
102                     if (self.__moveVertical == 'down'):
103                         yMove = (self.__startY - 1) % len(self.__inputImage[xMove])
104                     self.__startY = yMove
105
106                 x = 0
107                 while x < len(self.__outputImage):
108                     y = 0
109                     while y < len(self.__outputImage[x]):
110                         self.__outputImage[x][y] = self.__inputImage[xMove][yMove]
111                         y += 1
112                         yMove = (yMove + 1) % len(self.__inputImage[xMove])
113                     x += 1
114                     xMove = (xMove + 1) % len(self.__inputImage)
115                     yMove = self.__startY
116
117             time.sleep(self.__moveTime)

```

- Diese Methode bewegt das Bild nach rechts, links, oben, unten (inkl. Kombinationen davon). Dabei werden die Farbinformationen des Input-Bildes in das Output-Bild übertragen.
- Die Methode arbeitet nur, wenn der Move-Flag (`self.__move`) gesetzt ist. Daher müssen neue (Input-) Bilder (`setInputImage(self, image, imageFormat)`) von `_loadImage__(self)` ins Output-Bild geladen werden.
- Die Bildformat-Informationen unterdrücken gewisse Bewegungen:
 - Portrait (= Hochformat) unterdrückt die Bewegung nach links / rechts.
 - Landscape (= Querformat) unterdrückt die Bewegung nach oben / unten.
- Die Input-Bildgrösse ist für das Funktionieren der Methode nicht relevant.
Allerdings werden „zu kleine“ Bilder wiederholt bis sie die Grösse des Output-Bildes erreicht haben.

Public Methoden

#8 `finish(self)`

```
145     def finish(self):
146         """ Stoppt die Threads "moveImage" und "blink". """
147         # input("\nPress key to finish Thread\n")
148         self.__stopped = True
```

- Damit der Move- und der Blink-Thread beendet werden, `__moveImage__()` und `__blink__()`. Bei Bedarf kann die `input`-Methode verwendet werden, damit die Beendung der Threads erst mit der Tastatureingabe „Enter“ erfolgen. → `input("Press key to exit\n")`
- Im Testfile: # 8
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/finish [PUT]`

#9 `getMatrixPosition(self) → list`

```
150     def getMatrixPosition(self) -> list:
151         """ Gibt die Position zurück, wo das Bild auf dem LED-Display abgebildet werden soll.
152             [[Startwert x, Anzahl Spalten], [Startwert y, Anzahl Zeilen]]
153             :return lsMatrixValue: list
154         """
155         lsMatrixValue = [self.__matrix_xValues, self.__matrix_yValues]
156         return lsMatrixValue
```

- Gibt eine Liste zurück mit den Koordinaten der Bildposition im Gesamtbild auf dem LED-Display.
Listenformat: `[[x-Startwert, Anzahl Spalten], [y-Startwert, Anzahl Zeilen]]`
- Im Testfile: # 9
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/getMatrixPosition [GET]`

#10 `getXvalues(self) → list`

```
158     def getXvalues(self) -> list:
159         """ [Startwert x, Anzahl Spalten]
160             :return matrix_xValues: list
161         """
162         return self.__matrix_xValues
```

- Gibt eine Liste zurück mit den x-Koordinaten der Bildposition im Gesamtbild auf dem LED-Display.
Listenformat: `[x-Startwert, Anzahl Spalten]`
- Wird von der `LEDmatrix`-Klasse im `LEDmatrix_split_merge_171110.py` von der `__runMerge__()`-Methode verwendet.
- Im Testfile: # 10
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/getXvalues [GET]`

#11 `getYvalues(self) → list`

```
164     def getYvalues(self) -> list:
165         """ [Startwert y, Anzahl Zeilen]
166             :return matrix_yValues: list
167         """
168         return self.__matrix_yValues
```

- Gibt eine Liste zurück mit den y-Koordinaten der Bildposition im Gesamtbild auf dem LED-Display.
Listenformat: `[y-Startwert, Anzahl Zeilen]`
- Wird von der `LEDmatrix`-Klasse im `LEDmatrix_split_merge_171110.py` von der `__runMerge__()`-Methode verwendet.
- Im Testfile: # 11
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/getYvalues [GET]`

#39 `getName(self) → str`

```

415     def getName(self) -> str:
416         """ Gibt den Namen der Instanz zurück.
417         :return: objectName: str
418         """
419         return self.__objectName

```

- Gibt den Namen der Bildinstanz zurück.
- Im Testfile: # 39
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/getName [GET])`

===== Image- Methoden =====

#12 `getImage(self) → list`

```

173     def getImage(self) -> list:
174         """ Gibt das aktuelle Bild zurück (outputImage)
175         oder ein leeres Bild (lightOffImage)
176         wenn das Licht ausgeschalten ist.
177         """
178         if (self.__light is True):
179             return self.__outputImage
180         else:
181             return self.__lightOffImage
182

```

- Gibt ein Bild zurück (Liste mit RGB-Farbinformationen).
- Abhängig ob das Licht `self.__light` auf `True` (an) oder `False` (aus) gesetzt ist, wird entweder das `self.__outputImage` oder das `self.__lightOffImage` zurückgegeben.
- Im Testfile: # 12
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/getImage [GET])`

#13 `getLightOffImage(self) → list`

```

184     def getLightOffImage(self) -> list:
185         """ Gibt ein leeres Bild zurück, Farbwerte [0,0,0].
186         """
187         return self.__lightOffImage
188

```

- Gibt ein „leeres“ Bild in der Grösse der Bildinstanz zurück (Default-RGB-Farbinformationen = [0,0,0]).
- Im Testfile: # 13
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/getLightOffImage [GET])`

#14 setLightOffImage(self, image: list)

```

200     def setLightOffImage(self, image: list):
201         """ Definiert ein "Licht aus Bild".
202             :param image: list
203
204         x = 0
205         xImage = 0
206         yImage = 0
207         while x < len(self.__lightOffImage):
208             y = 0
209             while y < len(self.__lightOffImage[x]):
210                 self.__lightOffImage[x][y] = image[xImage][yImage]
211                 y += 1
212                 yImage = (yImage + 1) % len(image[xImage])
213             x += 1
214             xImage = (xImage + 1) % len(image)
215             yImage = 0

```

- Mit dieser Funktion kann das „leere“-„Licht aus“-Bild durch ein anderes Bild ersetzt werden.
- Das übergebene Bild, wird auf die Grösse der Bildinstanz angepasst bzw. in das bestehende *LightOffImage* eingelesen. **Das *LightOffImage* kann nicht bewegt werden, daher sollte das übergebene Bild der Instanzgrösse entsprechen!**
- Im Testfile: # 14
- REST-API:
`@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/setLightOffImage/<list:image> [PUT]`

#15 setLightOffImageColor(self, red: int, green: int, blue: int)

```

196     def setLightOffImageColor(self, red: int, green: int, blue: int):
197         """ Aendert die Farbe des "Licht aus Bildes". Int-Wertebereich von 0 bis 255.
198             :param red: int
199             :param green: int
200             :param blue: int
201
202         self.__lightOffImage = [[[red, green, blue] for i in range(self.__matrix_yValues[1])] for i in
203                               range(self.__matrix_xValues[1])]

```

- Diese Funktion passt die Farbe des „Licht aus“-Bildes an.
- Nur unifarben möglich. RGB-Werte von 0 bis 255.
- Im Testfile: # 15
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/setLightOffImageColor/
<int:red>/<int:green>/<int:blue> [PUT]`

#16 resetLightOffImage(self)

```

205     def resetLightOffImage(self):
206         """ Zurücksetzen des "Licht aus Bildes", mit den Farbwerten [0,0,0]. """
207         self.__lightOffImage = [[[0 for i in range(3)] for i in range(self.__matrix_yValues[1])] for i in
208                               range(self.__matrix_xValues[1])]

```

- Das „Licht aus“-Bild wird zurückgesetzt auf ein „leeres“ Bild mit den RGB-Farbwerthen (0,0,0).
- Im Testfile: # 16
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/resetLightOffImage [PUT]`

#17 setImageFormat(self, imageFormat: 'portrait' 'landscape' = None)

```

235     def setImageFormat(self, imageFormat: 'portrait' 'landscape' = None):
236         """ Mögliche Auswahl: 'default', 'portrait' (= hochformat), 'landscape' (= querformat)
237         Portrait-Bilder lassen sich nur nach oben/unten bewegen.
238         Landscape-Bilder lassen sich nur nach links/rechts bewegen
239         :param imageFormat: str
240
241         self.__imageFormat = imageFormat

```

- Mit dieser Funktion wird das Bildformat gesetzt: 'portrait' = hochformat, 'landscape' = querformat
- 'portrait' blockiert die links-/rechts-Bewegung, daher ist die Bewegung nur nach oben/unten möglich.
- 'landscape' blockiert die oben-/unten-Bewegung, daher ist die Bewegung nur nach links/rechts möglich.
- Default: None
- Im Testfile: # 17
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/setImageFormat/
<str:imageFormat> [PUT]

#18 setInputImage(self, image: list, imageFormat: 'portrait' 'landscape' = None)

```

244     def setInputImage(self, image: list, imageFormat: 'portrait' 'landscape' = None):
245         """ Ändert das input-Bild.
246         Die Bildgrösse sollte der Grösse des instanzierten Objektes MoveableImage() entsprechen.
247         Die optimale Bildgrösse kann mit getImageSizeAsText() bzw. mit getImageSize() abgefragt werden.
248         Ausserdem kann das Bildformat gewählt werden: default, portrait = hochformat, landscape = querformat.
249         Portrait-Bilder lassen sich nur nach oben/unten bewegen.
250         Landscape-Bilder lassen sich nur nach links/rechts bewegen
251         :param image: list
252         :param imageFormat: str
253
254         self.__ startX = 0
255         self.__ startY = 0
256         self.__ inputImage = image
257         self.__ loadImage_()
258         self.__ imageFormat = imageFormat

```

- Ändert das Input-Bild, welches als Ausgangslage für jegliche Bewegungen dient.
- Setzt die Variablen `self.__startX` und `self.__startY` zurück, welche für die Bildbewegung benötigt werden.
- Ruft `__loadImage_()` auf, um das neue Bild zu laden.
- Im Testfile: # 18
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/setInputImage/
<list:image>/<str:imageFormat> [PUT]

#19 getImageSizeAsText(self) → str

```

258     def getImageSizeAsText(self) -> str:
259         """ :return image-size: str
260
261         size = self.getImageSize()
262         return "Optimale Bildgrösse: x = %d, y = %d." % (size[0], size[1])

```

- Gibt einen Text zurück, welcher die Information über die optimale Bildgrösse enthält.
- Format: „Optimale Bildgrösse: x = ..., y =“
- Im Testfile: # 19
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/getImageSizeAsText [GET]

#20 getSize(self) → list

```

237     def getSize(self) -> list:
238         """ [x, y] | x = Anzahl Spalten / y = Anzahl Zeilen
239         |
240         :return image-size: list
241
242         return [self.__matrix_xValues[1], self.__matrix_yValues[1]]

```

- Gibt eine Liste zurück mit der Bildgrösse der Instanz.
- Format: [Anzahl Spalten (x), Anzahl Zeilen (y)]
- Im Testfile: # 20
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/getSize [GET])

===== Move - Methoden =====

#21 startBlink(self)

```

246     def startBlink(self):
247         """ Blink-Flag wird auf True gesetzt. """
248
249         self.__blinken = True

```

- Das Bild beginnt zu blinken.
- Im Testfile: # 21
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/startBlink [PUT])

#22 stopBlink(self)

```

282     def stopBlink(self):
283         """ Blink-Flag wird auf False gesetzt. """
284
285         self.__blinken = False
286         time.sleep(1)
287         self.__light = True

```

- Das Blinken wird gestoppt. Das Licht self.__light wird auf True (= Licht an) gesetzt.
- Scheint mit time.sleep() besser zu funktionieren.
- Im Testfile: # 22
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/stopBlink [PUT])

#23 startMove(self)

```

254     def startMove(self):
255         """ Move-Flag wird auf True gesetzt. """
256
257         self.__move = True

```

- Das Bild bewegt sich nur, wenn zusätzlich eine horizontale (right / left) oder vertikale (up / down) Bewegung gesetzt ist.
- Im Testfile: # 23
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/startMove [PUT])

#24 waiting(self)

```

258     def waiting(self):
259         """ Move-Flag wird auf False gesetzt. """
260
261         self.__move = False

```

- Die Bildbewegung wird gestoppt. Die Bewegungsinformationen bleiben erhalten.
- Im Testfile: # 24
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/waiting [PUT])

#25 stopMove(self)

```

299     def stopMove(self):
300         """ Move-Flag wird auf False gesetzt.
301             moveHorizontal, moveVertical werden auf 'false' gesetzt.
302             """
303         self.__move = False
304         self.__moveHorizontal = None
305         self.__moveVertical = None

```

- Die Bewegung wird gestoppt und die Bewegungsinformationen zurückgesetzt.
- Im Testfile: # 25
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/stopMove [PUT])

#26 resetMove(self)

```

308     def resetMove(self):
309         """ Ruft die Methoden stopMove() und stopBlink() auf.
310             startX und startY werden auf 0 gesetzt.
311             """
312         self.stopMove()
313         self.stopBlink()
314         self.__startX = 0
315         self.__startY = 0
316         self.__loadImage_()

```

- Die Bewegung wird gestoppt, die Bewegungsinformationen und die Bildposition werden zurückgesetzt.
- Das Input-Bild wird neu in der Ausgangsposition ins Output-Bild geladen.
- Achtung: self.__loadImage__() in dieser Methode wurde nicht getestet.
- Im Testfile: # 26
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/resetMove [PUT])

#27 moveRight(self)

```

279     def moveRight(self):
280         """ Ruft startMove() auf, moveHorizontal wird auf 'right' gesetzt. """
281         self.startMove()
282         self.__moveHorizontal = 'right'

```

- Die Bewegung nach rechts wird gestartet.
- Im Testfile: # 27
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/moveRight [PUT])

#28 moveLeft(self)

```

284     def moveLeft(self):
285         """ Ruft startMove() auf, moveHorizontal wird auf 'left' gesetzt. """
286         self.startMove()
287         self.__moveHorizontal = 'left'

```

- Die Bewegung nach links wird gestartet.
- Im Testfile: # 28
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/moveLeft [PUT])

#29 stopMoveHorizontal(self)

```
331     def stopMoveHorizontal(self):
332         """ moveHorizontal wird auf 'false' gesetzt. """
333         self.__moveHorizontal = None
```

- Die Information zur Bewegung nach links oder rechts wird zurückgesetzt.
- Im Testfile: # 29
- REST-API: @LEDmatrix.route('/ledmatrix.bfh.ch/api/v1.0/MoveableImage/stopMoveHorizontal [PUT])

#30 moveUp(self)

```
293     def moveUp(self):
294         """ Ruft startMove() auf, moveVertical wird auf 'up' gesetzt. """
295         self.startMove()
296         self.__moveVertical = 'up'
```

- Die Bewegung nach oben wird gestartet.
- Im Testfile: # 30
- REST-API: @LEDmatrix.route('/ledmatrix.bfh.ch/api/v1.0/MoveableImage/moveUp [PUT])

#31 moveDown(self)

```
298     def moveDown(self):
299         """ Ruft startMove() auf, moveVertical wird auf 'down' gesetzt. """
300         self.startMove()
301         self.__moveVertical = 'down'
```

- Die Bewegung nach unten wird gestartet
- Im Testfile: # 31
- REST-API: @LEDmatrix.route('/ledmatrix.bfh.ch/api/v1.0/MoveableImage/moveDown [PUT])

#32 stopMoveVertical(self)

```
348     def stopMoveVertical(self):
349         """ moveVertical wird auf 'false' gesetzt. """
350         self.__moveVertical = None
```

- Die Information zur Bewegung nach oben oder unten wird zurückgesetzt.
- Im Testfile: # 32
- REST-API: @LEDmatrix.route('/ledmatrix.bfh.ch/api/v1.0/MoveableImage/stopMoveVertical [PUT])

#33 getMoveInfo(self) → list

```

348     def getMoveInfo(self) -> list:
349         """ Gibt die Bewegungsinformationen zurück:
350             move, moveHorizontal, moveVertical, moveTime,
351             blinken, heartrate, Hz, timingRatio, on_off
352             :return lsMove: list
353
354         lsMove = [[['move', self.__move],
355                    ['horizontal', self.__moveHorizontal],
356                    ['vertical', self.__moveVertical],
357                    ['moveTime', self.__moveTime],
358                    ['blinken', self.__blinken],
359                    ['heartrate', self.__heartrate],
360                    ['rate', self.__Hz],
361                    ['timingRatio', self.__timingRatio],
362                    ['light on/light off', self.__on_off]]]
363
364     return lsMove

```

- Gibt eine Liste zurück mit sämtlichen Bewegungsinformationen (9 Parameter) des Bildes.
`move, moveHorizontal, moveVertical, moveTime, blinken, heartrate, Hz, timingRatio, on_off`
- Format:

```

[['move', self.__move],
 ['horizontal', self.__moveHorizontal],
 ['vertical', self.__moveVertical],
 ['moveTime', self.__moveTime],
 ['blinken', self.__blinken],
 ['heartrate', self.__heartrate],
 ['rate', self.__Hz],
 ['timingRatio', self.__timingRatio],
 ['light on/light off', self.__on_off]]

```

- Im Testfile: # 33
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/getMove [GET]')`

===== Geschwindigkeiten, Frequenzen - Methoden =====

Hypothese: on/off müssen jeweils $\geq 20\text{ms}$ sein (FrameDuration), Annahme Taktverhältnis = 1:1

Berechnung: $1000\text{ms} / (\text{FrameDuration ms} * (\text{Verhältnis on} + \text{Verhältnis off})) = \text{Anzahl Bilder/sec (Abgerundet)}$

Beispiele für die Berechnung der Maximalen Anzahl Bilder pro Sekunde in Abhängigkeit des Taktverhältnisses (Timing Ratio) und einer Frame-Duration von 20ms:

- Timing Ratio = 1:1 → $1000\text{ms}/(20\text{ms} * 2) = 1000\text{ms}/40\text{ms} \rightarrow \leq 25 \text{ Bilder/sec}$
- Timing Ratio = 1:2 → $1000\text{ms}/(20\text{ms} * 3) = 1000\text{ms}/60\text{ms} \rightarrow \leq 16 \text{ Bilder/sec}$
- Timing Ratio = 1:3 → $1000\text{ms}/(20\text{ms} * 6) = 1000\text{ms}/80\text{ms} \rightarrow \leq 12 \text{ Bilder/sec}$

#34 setMoveTime(self, second: float = 1.0)

```

368     def setMoveTime(self, second: float = 1.0):
369         """ Beeinflusst die time.sleep() beim moveImage-Thread.
370             Default: 1 Sekunde.
371             :param second: float
372
373         self.__moveTime = second

```

- Setzt die Bewegungsgeschwindigkeit. Das Bild wird alle x-Sekunden bewegt. Default = 1 Sekunde.
- Im Testfile: # 34
- REST-API:
`@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/setMoveTime/<float:second> [PUT]')`

#35 setRate(self, hertz: float = 1.0)

```

376     def setRate(self, hertz: float = 1.0):
377         """ Anzahl Vorgänge pro Sekunde, diese Methode nicht für Puls-Frequenzen aufrufen!
378             Berechnet die Light-on/off-Zeit unter Berücksichtigung des Heartrate-Flags.
379             :param hertz: float
380
381             self.__Hz = hertz
382             if (self.__heartrate is False):
383                 self.__on_off[0] = (1 / (hertz * (self.__timingRatio[0] + self.__timingRatio[1]))) * self.__timingRatio[0]
384                     # on
385                 self.__on_off[1] = (1 / (hertz * (self.__timingRatio[0] + self.__timingRatio[1]))) * self.__timingRatio[1]
386                     # off
387             else:
388                 if ((1 / self.__Hz) - self.__heartrateOff > 0):
389                     self.__on_off[0] = (1 / self.__Hz) - self.__heartrateOff
390                     self.__on_off[1] = self.__heartrateOff
391             else:
392                 x = 1 / (self.__Hz * 2)
393                 self.__on_off[0] = x
394                 self.__on_off[1] = x

```

- Setzt die Blink-Frequenz. Die Frequenz in Hertz, gibt die Anzahl Vorgänge pro Sekunde an.
Default = 1 Ereignis/sec.
- Die Methode berechnet anhand der timing-Ration oder des gesetzten heartrate-Flags, die Dauer von Licht an und Licht aus.
- Im Testfile: # 35
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/setRate/<float:hertz> [PUT])

#36 setPuls(self, puls: int)

```

346     def setPuls(self, puls: int):
347         """ Puls übergeben. Methode setzt den Heartrate-Flag auf True (default-Wert).
348             Ruft setRate() und startBlink() auf.
349             :param puls: int
350
351             self.setRate(puls / 60)
352             self.setHeartrateFlag()
353             self.startBlink()

```

- Mit dieser Methode kann die Pulsfrequenz übermittelt werden.
- Die Methode berechnet daraus die Frequenz pro Sekunde (Hertz).
Berechnung: Gemessener Puls / 60 = Puls pro Sekunde
- Sie setzt den heartrate-Flag und startet das Blinken.
- Im Testfile: # 36
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/setPuls/<int:puls> [PUT])

#37 setTimingRatio(self, ratioOn: int, ratioOff: int)

```

407     def setTimingRatio(self, ratioOn: int = 1, ratioOff: int = 1):
408         """ Ändert das Taktverhältnis der Blink-Frequenz.
409             Defaultmäßig ist [1,1] definiert, also gleich lange Licht ein wie aus.
410             :param ratioOn:
411             :param ratioOff:
412
413             self.__timingRatio[0] = ratioOn
414             self.__timingRatio[1] = ratioOff
415             if (self.__heartrate is False):
416                 self.__on_off[0] = (1 / (self.__Hz * (ratioOn + ratioOff))) * ratioOn # on
417                 self.__on_off[1] = (1 / (self.__Hz * (ratioOn + ratioOff))) * ratioOff # off

```

- Hier kann das Taktverhältnis der Blinkfrequenz eingestellt werden. (in ganzen Zahlen angeben)
- Default: je 1 => 1:1, also das Licht ist gleich lang an wie aus.
- Im Testfile: # 37
- REST-API: @LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/setTimingRatio/
<int:ratioOn>/<int:ratioOff> [PUT])

#38 setHeartrateFlag(self, boolean: bool = True)

```

367     def setHeartrateFlag(self, boolean: bool = True):
368         """ wird der Flag auf true gesetzt,
369         wird der timme.sleep-Wert (self.__on_off) mit setRate() anders berechnet
370         :param boolean: bool
371
372         if (boolean is True):
373             self.__heartrate = True
374         else:
375             self.__heartrate = False
376             self.setRate(self.__Hz)

```

- Damit wird bei `setRate()` die `self.__on_off` anders berechnet, mit einer fixen `heartrateOff`-Zeit. `self.__heartrateOff` beträgt defaultmässig 0.5 Sekunden. Da es sich nur um eine „protected“-Variable handelt, kann von aussen darauf zugegriffen werden.
- Im Testfile: # 38
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/setHeartrate/ <bool:boolean> [PUT]`

#40 light(self, state: bool = True)

```

437     def light(self, state: bool = True):
438         """ Um das Licht ein- und auszuschalten.
439         :param state: bool
440
441         self.__blitzen = False
442         time.sleep(1)
443         self.__light = state

```

- Um das Licht „ein“ oder „aus“ auf der Ebene der Bildinstanz zu steuern.
- Im Testfile: # 40
- REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/MoveableImage/light/<bool:state> [GET]`

Testfile: Test_moveableImage_171110 Letztmals erfolgreich: 01.12..2017

- Test_moveableImage_171110_v1.py
- Test_moveableImage_171110_v2.py → In diesem Kapitel beschrieben.
- Test_moveableImage_171110_v3.py

Um die Bildbewegungen zu testen, werden die Klassen *TestThread* und *Merge* benötigt, welche die geänderten Bilder an die LEDmatrixConnect-Instanz übermitteln.

Imports:

- from LEDmatrix_init_171110 import *
- from LEDmatrix_moveableImage_171110 import *
- import imageProperties
- import time
- import threading

Klasse: TestThread

```

9  class TestThread:
10     """ Klasse für die automatische Bildübermittlung. """
11
12     def __init__(self, matrixObject: LEDmatrixConnect, imageObject: MoveableImage):
13         self.matrix = matrixObject
14         self.image = imageObject
15         self.xValues = self.image.getXvalues()
16         self.yValues = self.image.getYvalues()
17
18         # Thread
19         self.stopped = False
20         self.sleepTime = 0.01
21
22         self.threadTest = threading.Thread(target=self.__run__, args=())
23         self.threadTest.daemon = True
24         self.threadTest.start()
25
26     def __run__(self):
27         """ Thread, welcher die Bildteile zusammenfügt und an die LEDmatrix_init weiterleitet. """
28         while True:
29             if (self.stopped is True):
30                 print("Test-Thread stopped\n")
31                 break
32
33             mergeImage = self.image.getImage()
34             matrixImage = [[[0, 0, 0] for i in range(10)] for i in range(20)]
35
36             xVal = self.xValues
37             yVal = self.yValues
38
39             x = 0
40             y = 0
41             while x < xVal[1]:
42                 while y < yVal[1]:
43                     matrixImage[xVal[0] + x][yVal[0] + y] = mergeImage[x][y]
44                     y += 1
45                 x += 1
46                 y = 0
47
48             self.matrix.setImage(matrixImage)
49             time.sleep(self.sleepTime)

```

- Um die geänderten Output-Bilder automatisch an die LEDmatrixConnect-Instanz zu übermitteln.
- Da es sich um ein Testprogramm handelt sind alle Variablen dieser Klasse public.

Klasse: Merge

```
52     class Merge:
53         """ Klasse um die Bildübermittlung manuell aufzurufen. """
54
55     def __init__(self, matrixObject: LEDmatrixConnect, imageObject: MoveableImage):
56         self.matrix = matrixObject
57         self.image = imageObject
58         self.xValues = self.image.getXvalues()
59         self.yValues = self.image.getYvalues()
60
61
62     def run(self, image: list):
63         """ Bildübergabe an LEDmatrixConnect-Objekt. """
64         mergeImage = image
65         matrixImage = [[[0, 0, 0] for i in range(10)] for i in range(20)]
66
67         xVal = self.xValues
68         yVal = self.yValues
69
70         x = 0
71         y = 0
72         while x < xVal[1]:
73             while y < yVal[1]:
74                 matrixImage[xVal[0] + x][yVal[0] + y] = mergeImage[x][y]
75                 y += 1
76             x += 1
77             y = 0
78
79         self.matrix.setImage(matrixImage)
80         time.sleep(1.5)
```

- Um die geänderten Output-Bilder „manuell“ an die LEDmatrixConnect-Instanz zu übermitteln.
- Da es sich um ein Testprogramm handelt sind alle Variablen sowie die Methode dieser Klasse public.

Ausführendes Testprogramm:

```

85 ► if __name__ == "__main__":
86     # LED-Display-Verbindung und Bildinstanz erstellen
87     matrix = LEDmatrixConnect()
88     # Erwartung: ein 10x mal 6y grosses Bild in der Mitte des Displays
89     image = MoveableImage('test', imageProperties.forms1, [0, 20], [0, 10])
90
91     # Um die Bilder zu übermitteln
92     send = Merge(matrix, image)
93
94     # Abfragen
95     print('\ngetName: ' + image.getName())                                # 39
96     print('\ngetMatrixPosition (Expected: [[0, 20][0, 10]]): ', image.getMatrixPosition()) # 9
97     print('\ngetXvalues (Expected: [0, 20]): ', image.getXvalues())           # 10
98     print('\ngetYvalues (Expected: [0, 10]): ', image.getYvalues())           # 11
99     print('\ngetImageSizeAsText: ' + image.getImageSizeAsText())           # 19
100    print('\ngetImageSize: ', image.getImageSize())                         # 20
101    print('\ngetMoveInfo (1): \n', image.getMoveInfo())                      # 33
102

```

- ✓ getName: test
- ✓ getMatrixPosition (Expected: [[0, 20][0, 10]]): [[0, 20], [0, 10]]
- ✓ getXvalues (Expected: [0, 20]): [0, 20]
- ✓ getYvalues (Expected: [0, 10]): [0, 10]
- ✓ getImageSizeAsText: Optimale Bildgrösse: x = 20, y = 10.
- ✓ getImageSize: [20, 10]
- ✓ getMoveInfo (1):
 - [['move', False],
 - ['horizontal', None],
 - ['vertical', None],
 - ['moveTime', 1],
 - ['blinken', False],
 - ['heartrate', False],
 - ['rate', 1],
 - ['timingRatio', [1, 1]],
 - ['light on/light off', [0.5, 0.5]]]

```

104     print('\nBild-Tests: Image ändern, "Licht aus"-Image ändern')
105     send.run(image.getImage())                                         # 12
106     time.sleep(2)
107     send.run(image.getLightOffImage())                                    # 13
108     image.setLightOffImageColor(0, 255, 0)                            # 15
109     time.sleep(2)
110     send.run(image.getLightOffImage())                                    # 13
111     image.setLightOffImage(imageProperties.legoface)                  # 14
112     time.sleep(2)
113     send.run(image.getLightOffImage())                                    # 13
114     image.resetLightOffImage()                                       # 16
115     time.sleep(2)
116     send.run(image.getLightOffImage())                                    # 13
117     image.setInputImage(imageProperties.hearts1, 'landscape')        # 18
118     time.sleep(2)
119     send.run(image.getImage())                                         # 12
120     time.sleep(1)

```

- ✓ Bild-Tests: Image ändern, "Licht aus"-Image ändern

```

122      # Thread starten, um die Bildbewegungen zu testen
123      TestThread(matrix, image)
124
125      print("\nBild: Blink- und Move-tests")
126      image.startBlink()                                     # 21 blinken
127      time.sleep(2)
128      image.setRate(4)                                      # 35 4x/Sekunde blinken
129      image.setTimingRatio(3, 1)                            # 37 im Verhältnis 3:1
130      print('\ngetMoveInfo (2): \n', image.getMoveInfo())
131      image.setHeartrateFlag()                           # 33
132      print('\ngetMoveInfo (3): \n', image.getMoveInfo())
133      image.setPuls(120)                                    # 36 Puls: 120
134      print('\ngetMoveInfo (4): \n', image.getMoveInfo())
135      image.setHeartrateFlag(False)                      # 38 Heartrate: False
136      print('\ngetMoveInfo (5): \n', image.getMoveInfo())
137      image.stopBlink()                                   # 22 blinken stoppen
138      time.sleep(2)

```

- ✓ Bild: Blink- und Move-tests

- | | |
|---|---|
| <ul style="list-style-type: none"> ✓ getMoveInfo (2):
 [['move', False],
 ['horizontal', None],
 ['vertical', None],
 ['moveTime', 1],
 ['blinken', True],
 ['heartrate', False],
 ['rate', 4],
 ['timingRatio', [3, 1]],
 ['light on/light off', [0.1875, 0.0625]]] | <ul style="list-style-type: none"> ✓ getMoveInfo (4):
 [['move', False],
 ['horizontal', None],
 ['vertical', None],
 ['moveTime', 1],
 ['blinken', True],
 ['heartrate', True],
 ['rate', 2.0],
 ['timingRatio', [3, 1]],
 ['light on/light off', [0.25, 0.25]]] |
| <ul style="list-style-type: none"> ✓ getMoveInfo (3):
 [['move', False],
 ['horizontal', None],
 ['vertical', None],
 ['moveTime', 1],
 ['blinken', True],
 ['heartrate', True],
 ['rate', 4],
 ['timingRatio', [3, 1]],
 ['light on/light off', [0.1875, 0.0625]]] | <ul style="list-style-type: none"> ✓ getMoveInfo (5):
 [['move', False],
 ['horizontal', None],
 ['vertical', None],
 ['moveTime', 1],
 ['blinken', True],
 ['heartrate', False],
 ['rate', 2.0],
 ['timingRatio', [3, 1]],
 ['light on/light off', [0.375, 0.125]]] |

```

140   print("\nimageFormat-Tests")
141   image.setImageFormat()
142   image.moveRight()
143   time.sleep(5)                                # 17 Bildformat 'default'
144   image.setMoveTime(0.1)                         # 27 nach rechts
145   time.sleep(2)
146   image.setMoveTime(0.5)                         # 34 schneller
147   time.sleep(2)
148   image.setMoveTime(0.1)                         # 34 langsamer
149   time.sleep(2)
150   image.moveLeft()                            # 28 nach links
151   print('\ngetMoveInfo (6): \n', image.getMoveInfo())
152   time.sleep(2)
153   image.setImageFormat('portrait')            # 33
154   time.sleep(2)
155   image.setImageFormat('landscape')           # 17 Bildformat: hoch (stop)
156   time.sleep(2)
157   image.setImageFormat('default')              # 17 Bildformat: quer
158   time.sleep(2)
159   image.stopMoveHorizontal()                  # 17 Bildformat: default
160   print('\ngetMoveInfo (7): \n', image.getMoveInfo())
161   time.sleep(2)
162   image.moveUp()                             # 29 links/rechts stoppen
163   time.sleep(2)
164   image.moveDown()                           # 30 nach oben
165   time.sleep(2)
166   image.setImageFormat('landscape')          # 31 nach unten
167   time.sleep(2)
168   image.setImageFormat('portrait')            # 17 Bildformat: quer (stop)
169   time.sleep(2)
170   image.setImageFormat('default')             # 17 Bildformat: hoch
171   time.sleep(2)
172   image.stopMoveVertical()                  # 17 Bildformat: default
173   time.sleep(2)                            # 32 oben/unten stoppen

```

✓ imageFormat-Tests

✓ getMoveInfo (6):

```

[['move', True],
 ['horizontal', 'left'],
 ['vertical', None],
 ['moveTime', 0.1],
 ['blinken', False],
 ['heartrate', False],
 ['rate', 2.0],
 ['timingRatio', [3, 1]],
 ['light on/light off', [0.375, 0.125]]]

```

✓ getMoveInfo (7):

```

[['move', True],
 ['horizontal', None],
 ['vertical', None],
 ['moveTime', 0.1],
 ['blinken', False],
 ['heartrate', False],
 ['rate', 2.0],
 ['timingRatio', [3, 1]],
 ['light on/light off', [0.375, 0.125]]]

```

```

175     print("\nstop and go")
176     image.moveRight()                                # 27 nach rechts
177     image.moveUp()                                 # 30 nach oben
178     time.sleep(3)
179     image.waiting()                               # 24 warten
180     time.sleep(2)
181     image.startMove()                            # 23 Bewegung starten
182     time.sleep(2)
183     image.stopMove()                            # 25 Bewegung stoppen
184     time.sleep(2)
185     image.startMove()                            # 23 "no effect"
186     print('\ngetMoveInfo (8): \n', image.getMoveInfo()) # 33
187     time.sleep(2)
188     image.resetMove()                           # 26 Bildposition zurück
189     time.sleep(2)

190
191     print("\nmove in alle Richtungen")
192     image.moveRight()                            # 27 rechts
193     time.sleep(1)
194     image.moveDown()                            # 31 unten (rechts + unten)
195     time.sleep(1)
196     image.moveLeft()                            # 28 links (links + unten)
197     time.sleep(1)
198     image.moveUp()                             # 30 oben (links + oben)
199     time.sleep(1)
200     image.resetMove()                           # 26 Bildposition zurück
201     time.sleep(2)

202
203     print("Light on / off")
204     image.startBlink()                           # 21 blinken
205     time.sleep(2)
206     image.light(False)                          # 40 Licht aus
207     time.sleep(2)
208     image.light()                             # 40 Licht ein
209     time.sleep(2)

```

- ✓ stop and go
- ✓ getMoveInfo (8):


```
[['move', True],
['horizontal', None],
['vertical', None],
['moveTime', 0.1],
['blinken', False],
['hearrate', False],
['rate', 2.0],
['timingRatio', [3, 1]],
['light on/light off', [0.375, 0.125]]]
```

- ✓ move in alle Richtungen
- ✓ Light on / off

```

211     # Alle Threads beenden, Verbindungsabbau
212     TestThread.stopped = True
213     image.finish()                                # 8
214     matrix.light(False)                         # 43
215     time.sleep(3)
216     matrix.finish()                            # 2

```

- ✓ Thread blink test stopped.
- ✓ Thread move test stopped.
- ✓ Verbindung beendet.

LEDmatrix_split_merge_171110.py: LEDmatrix()

Klasse:

LEDmatrix(self, usecase: 'single' 'square' 'colorHeartrate' 'quarters4' 'row2' 'col2' = 'single')

REST-API: `@LEDmatrix.route('ledmatrix.bfh.ch/api/v1.0/LEDmatrix [POST]')`

Imports:

- from LEDmatrix_init_171110.py import *
- from LEDmatrix_moveableImage_171110.py import *
- import imageProperties
- import matrixProperties
- import threading
- import time

Instanzvariablen

Attributname	Zuordnung	Beschreibung
self.__bgRed <i>Datentyp: int</i>	0	Definiert den Rotanteil der Hintergrundfarbe. Wertebereich 0 bis 255
self.__bgGreen <i>Datentyp: int</i>	0	Definiert den Grünanteil der Hintergrundfarbe. Wertebereich 0 bis 255
self.__bgBlue <i>Datentyp: int</i>	0	Definiert den Blauanteil der Hintergrundfarbe. Wertebereich 0 bis 255
self.__case <i>Datentyp: str</i>	UseCase: ➤ 'single' = default ➤ 'square' ➤ 'colorHeartrate' ➤ 'quarters4' ➤ 'row2' ➤ 'col2'	Speichert den gewählten UseCase.
self.__objectlist <i>Datentyp: list</i>	[]	Diese Liste enthält nach der Initialisierung alle Bildinstanzen = MoveableImage-Objekte.
self.__stopped <i>Datentyp: bool</i>	False	Varianten: False / True Wird von <code>__runMerge__()</code> benötigt.
self._sleepTime <i>Datentyp: float (Sekunden)</i> PROTECTED (es kann von aussen zugegriffen werden)	0.01	Hypothese: Die sleepTime des Merge-Threads sollte kleiner sein als die FrameDuration der <code>LEDmatrixConnect</code> -Klasse. Die Default-FrameDuration beträgt 20ms = 0.02 sec

Objekte

Attributname	Zuordnung	Beschreibung
self.__ledMatrix <i>Datentyp: LEDmatrixConnect</i>	<code>LEDmatrixConnect()</code>	Für die Verbindung zum LED-Display.
self.__threadMerge <i>Datentyp: undefined</i>	<code>threading.Thread(target=self.__runMerge__, args=())</code>	Um die Teilbilder zu einem Gesamtbild zusammenzufügen, welches der Grösse des LED-Displays entspricht.

Private Methoden

__init__(self) => Konstruktor

Enthält die oben aufgeführten Instanzvariablen und Objekte. Diese Methode dient der Initialisierung. Je nach Case werden entsprechende Bildinstanzen (MoveableImage) erstellt.

Case 'square'

```

31   if (self.__case == 'square'):
32       # Eine quadratische Bildinstanz in der Mitte.
33       squareImage = MoveableImage('square', [[[255, 255, 255] for i in range(10)] for i in range(10)], [5, 10],
34                                         [0, 10])
35       self.__objectlist.append(squareImage)

```

Case 'colorHeartRate'

```

38   if (self.__case == 'colorHeartrate'):
39       # Es werden 5 bewegbare Bildinstanzen erstellt.
40       # (oben = "Puls", unten-links = gelb, unten-links = blau, unten-3.v.links = weiss, unten-4.v.links = ).
41
42       # Default-Bilder importieren
43       text = imageProperties.pulstext
44       heart1 = imageProperties.heartblue
45       heart2 = imageProperties.heartred
46       heart3 = imageProperties.heartgreen
47       heart4 = imageProperties.heartyellow
48
49       # Matrix splitten
50       topImage = MoveableImage("topImage", text, [0, 20], [0, 5])
51       bottomLeftImage = MoveableImage("bottomLeftImage", heart1, [0, 5], [5, 5])
52       bottom2ndLeftImage = MoveableImage("bottom2ndLeftImage", heart2, [5, 5], [5, 5])
53       bottom3rdLeftImage = MoveableImage("bottom3rdLeftImage", heart3, [10, 5], [5, 5])
54       bottom4thLeftImage = MoveableImage("bottom4thLeftImage", heart4, [15, 5], [5, 5])
55       self.__objectlist.extend(
56           | (topImage, bottomLeftImage, bottom2ndLeftImage, bottom3rdLeftImage, bottom4thLeftImage))
57
58       # Definieren, dass der Puls dargestellt werden soll.
59       bottomLeftImage.setHeartRateFlag(True)
60       bottom2ndLeftImage.setHeartRateFlag(True)
61       bottom3rdLeftImage.setHeartRateFlag(True)
62       bottom4thLeftImage.setHeartRateFlag(True)

```

Case 'quarters4'

```

65   if (self.__case == 'quarters4'):
66       # Es werden 4 bewegbare Bildinstanzen erstellt.
67       # (oben-links = rot, oben-rechts = gelb, unten-links = blau, unten-rechts = weiss).
68       topLeftImage = MoveableImage("topLeftImage", [[[255, 0, 0] for i in range(5)] for i in range(10)], [0, 10],
69                                         [0, 5])
70       topRightImage = MoveableImage("topRightImage", [[[255, 255, 0] for i in range(5)] for i in range(10)],
71                                         [10, 10], [0, 5])
72       bottomLeftImage = MoveableImage("bottomLeftImage", [[[0, 0, 255] for i in range(5)] for i in range(10)],
73                                         [0, 10], [5, 5])
74       bottomRightImage = MoveableImage("bottomRightImage",
75                                         [[[255, 255, 255] for i in range(5)] for i in range(10)], [10, 10], [5, 5])
76       self.__objectlist.extend((topLeftImage, topRightImage, bottomLeftImage, bottomRightImage))

```

Case 'row2'

```

79   if (self.__case == 'row2'):
80       # Es werden 2 bewegbare Bildinstanzen erstellt.
81       # (obere und untere Bildinstanz). Farbe oben = rot, Farbe unten = blau.
82       topImage = MoveableImage("topImage", [[[255, 0, 0] for i in range(5)] for i in range(20)], [0, 20], [0, 5])
83       bottomImage = MoveableImage("bottomImage", [[[0, 0, 255] for i in range(5)] for i in range(20)], [0, 20],
84                                         [5, 5])
85       self.__objectlist.extend((topImage, bottomImage))

```

Case 'col2'

```

88   if (self.__case == 'col2'):
89       # Es werden 2 bewegbare Bildinstanzen erstellt.
90       # (linke und rechte Bildinstanz). Farbe links = rot, Farbe rechts = blau.
91       leftImage = MoveableImage("leftImage", [[[255, 0, 0] for i in range(10)] for i in range(10)], [0, 10],
92                                         [0, 10])
93       rightImage = MoveableImage("rightImage", [[[0, 0, 255] for i in range(10)] for i in range(10)], [10, 10],
94                                         [0, 10])
95       self.__objectlist.extend((leftImage, rightImage))

```

Case 'single' = Default Case

```

98     if (self.__case == 'single'):
99         # Die Matrix wird nicht geteilt, es wird nur 1 bewegbare Bildinstanz erstellt.
100        singleImage = MoveableImage("singleImage")
101        self.__objectlist.append(singleImage)

```

Aufruf	Beschreibung
self.__threadMerge.daemon	True
self.__threadMerge.start()	

__runMerge__(self) => Konstruktor

Enthält die oben aufgeführten Instanzvariablen und Objekte. Diese Methode dient der Initialisierung.

```

98     def __runMerge__(self):
99         """ Thread, welcher die Bildteile zusammenfügt und an die LEDmatrix_init weiterleitet. """
100        while True:
101            if (self.__stopped is True):
102                print("Merge-Thread stopped\n")
103                break
104
105            mergeImage = []
106            # image = [[[self.__bgRed, self.__bgGreen, self.__bgBlue] for i in range(10)] for i in range(20)]
107            image = [[[self.__bgRed, self.__bgGreen, self.__bgBlue] for i in range(mergeProperties.ROWS)] for i in
108                     range(mergeProperties.COLUMNS)]
109
110            # alle aktuellen Bilder der Bildinstanzen "abholen"
111            j = 0
112            while j < len(self.__objectlist):
113                mergeImage.append(self.__objectlist[j].getImage())
114                j += 1
115
116            # einzelne Bildteile zu einem Gesamtbild (20x/10y) zusammenfügen
117            i = 0
118            while i < len(self.__objectlist):
119                xValues = self.__objectlist[i].getXvalues()
120                yValues = self.__objectlist[i].getYvalues()
121
122                x = 0
123                y = 0
124                while x < xValues[1]:
125                    while y < yValues[1]:
126                        image[xValues[0] + x][yValues[0] + y] = mergeImage[i][x][y]
127                        y += 1
128                    x += 1
129                    y = 0
130                i += 1
131
132            self.__ledMatrix.setImage(image) # Gesamtbild an LEDmatrixConnect-Instanz
133            time.sleep(self.__sleepTime)

```

- Diese Methode fügt die einzelnen Bildteile zu einem Gesamtbild zusammen, welches der Display-Grösse entspricht. Dieses Bild wird an die LEDmatrixConnect-Instanz übergeben.
- Zeilen 107 und 108 mit mergeProperties.ROWS und mergeProperties.COLUMN sind ungetestet, getestet wurde mit der Zeile 106.

Public Methoden

#41 setBGcolor(self, red: int, green: int, blue: int)

```

139     def setBGcolor(self, red: int, green: int, blue: int):
140         """ Definiert die Hintergrundfarbe, wo kein Bild ist.
141         Wertebereich von 0 bis 255.
142         :param red: int
143         :param green: int
144         :param blue: int
145
146         self.__bgRed = red
147         self.__bgGreen = green
148         self.__bgBlue = blue

```

- Mit dieser Methode kann die Hintergrundfarbe des Gesamtbildes definiert werden. Sie kommt allerdings nur zur Geltung wenn die Teilbilder zusammen kleiner sind als das Gesamtbild von 20x/10y.
- RGB-Wertebereich: 0 bis 255
- Im Testfile: # 41
- REST-API: @LEDmatrix.route('/ledmatrix.bfh.ch/api/v1.0/LEDmatrix/setBGcolor/ <int:red>/<int:green>/<int:blue> [PUT])

#42 getObjects(self) → list

```

151     def getObjects(self) -> list:
152         """ Gibt eine Liste mit den Bildobjekten (MoveableImage) zurück.
153         :return: objectList: list
154
155         return self.__objectlist

```

- Gibt eine Liste zurück mit den Bildinstanzen, die für den gewählten Use-Case erstellt wurden.
- Im Testfile: # 42
- REST-API: @LEDmatrix.route('/ledmatrix.bfh.ch/api/v1.0/LEDmatrix/getObjects [GET])

#43 getMatrix(self) → LEDmatrixConnect

```

158     def getMatrix(self) -> LEDmatrixConnect:
159         """ Gibt das Matrix-Objekt (LEDmatrixConnect) zurück.
160         :return: ledMatrix: LEDmatrixConnect
161
162         return self.__ledMatrix

```

- Gibt die LEDmatrixConnect-Instanz zurück.
- Im Testfile: # 43
- REST-API: @LEDmatrix.route('/ledmatrix.bfh.ch/api/v1.0/LEDmatrix/getMatrix [GET])

#45 getCase(self) → str

```

170     def getCase(self) -> str:
171         """ Gibt den Namen des Use-Cases zurück.
172         :return: case: str
173
174         return self.__case

```

- Gibt den String des Use-Case-Namens zurück.
- Im Testfile: # 45
- REST-API: @LEDmatrix.route('/ledmatrix.bfh.ch/api/v1.0/LEDmatrix/getCase [GET])

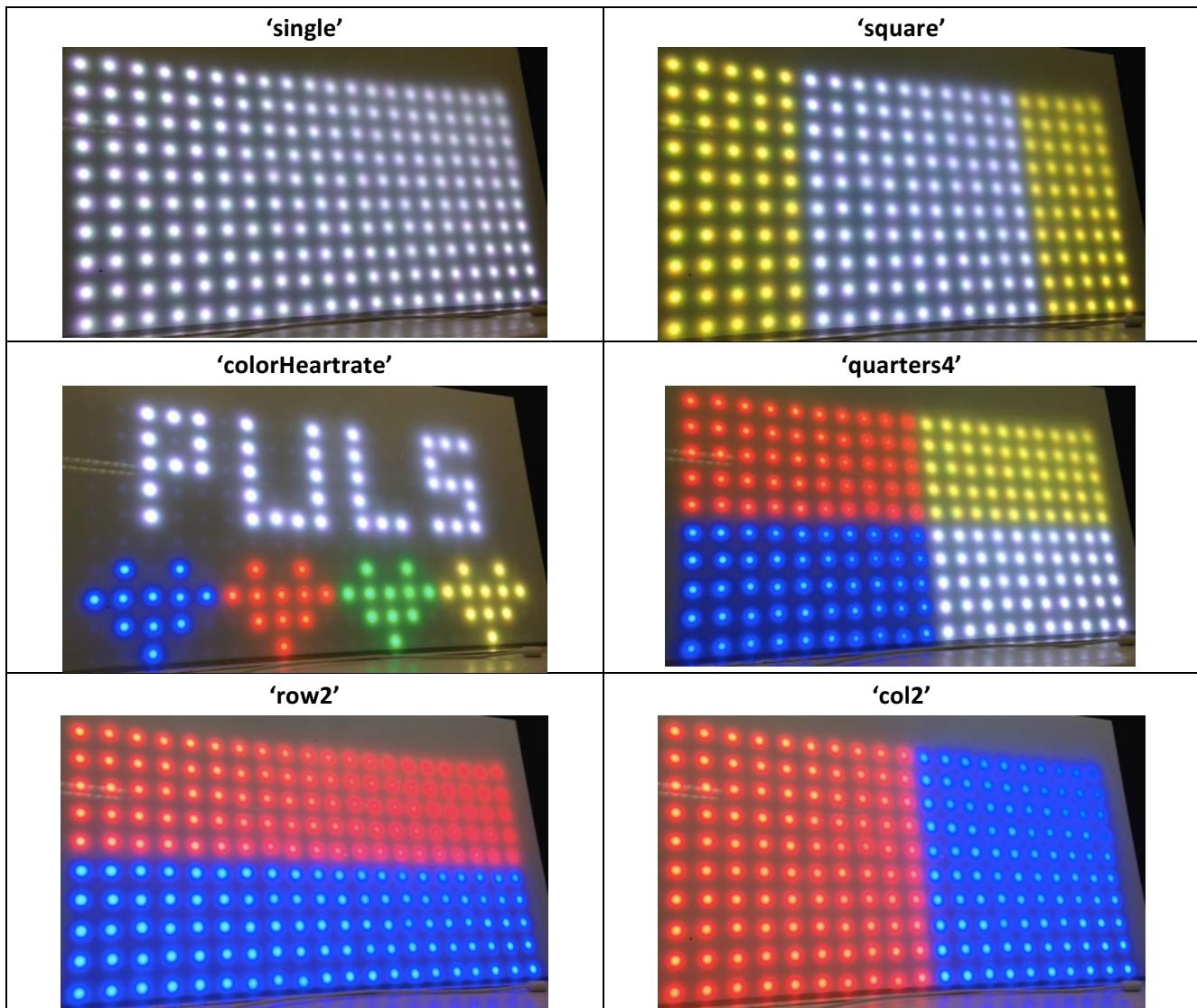
#44 *finish(self)*

```

165     def finish(self):
166         """ Programm beenden.
167             Beendet alle Threads und die Verbindung zum LED-Display.
168         """
169         objectlist = self.__objectlist
170
171         input("\nPress key to finish Thread\n")
172         self.__stopped = True
173
174         i = 0
175         while i < len(objectlist):
176             objectlist[i].finish()
177             i += 1
178
179         time.sleep(2)
180         input("\nPress key to exit\n")
181         self.__ledMatrix.finish()

```

- Nach diesem Methodenaufruf kann das Programm beendet werden.
Die Methode beendet nach „Enter“-Eingabe sämtliche Threads der Bildinstanzen, sowie den Merge-Thread. Nach einer erneuten „Enter“-Eingabe wird die Verbindung zum LED-Display beendet.
- Im Testfile: # 44
- REST-API: `@LEDmatrix.route('/ledmatrix.bfh.ch/api/v1.0/LEDmatrix/finish [PUT]`)

Darstellung Use Cases

Testfile: Test_split_merge_171110 Letztmals erfolgreich: 30.11.2017**Case 0: LEDmatrix()**

```
6      print("\nTest: Case 0 'default'\n")  # =====
7      matrix0 = LEDmatrix()
8      images0 = matrix0.getObjects()           # 42
9
10     i = 0
11     while (i < len(images0)):
12         print('\n%s-index-Nr. %d, Name: %s, Position: ' % (matrix0.getCase(), i, images0[i].getName()),
13             images0[i].getMatrixPosition())      # 45, 39, 9
14         i += 1
15
16     time.sleep(2)
17
18     matrix0.getMatrix().light(False)        # 43
19     time.sleep(3)
20     matrix0.getMatrix().light(True)         # 43
21     time.sleep(3)
22     matrix0.getMatrix().light(False)        # 43
23     time.sleep(3)
24
25     matrix0.finish()                      # 44
26     print('\nCase 0 Default(=single)-Test erfolgreich.\n')
27     time.sleep(5)
```

Ergebnis Case 0: LEDmatrix()

- ✓ Test: Case 0 'default'
- ✓ Instanz LEDmatrix wurde erstellt.
- ✓ Verbindung zu brickd hergestellt.
- ✓ Callback aktiviert.
- ✓ Default-index-Nr. 0, Name: singleImage, Position: [[0, 20], [0, 10]]
- ✓ Press key to finish Thread
- ✓ Merge-Thread stopped
- ✓ Thread blink singleImage stopped.
- ✓ Thread move singleImage stopped.
- ✓ Press key to exit
- ✓ Verbindung beendet.
- ✓ Case 0 Default(=single)-Test erfolgreich.

Case 1: LEDmatrix('square')

```
29      print("\nTest: Case 1 'square'\n")  # =====
30      matrix1 = LEDmatrix('square')
31      images1 = matrix1.getObjects()           # 42
32
33      i = 0
34      while (i < len(images1)):
35          print('\n%s-index-Nr. %d, Name: %s, Position: ' % (matrix1.getCase(), i, images1[i].getName()),
36                images1[i].getMatrixPosition())       # 45, 39, 9
37          i += 1
38
39      time.sleep(2)
40      matrix1.setBGcolor(255, 0, 255)         # 41
41      time.sleep(2)
42      matrix1.setBGcolor(255, 255, 0)         # 41
43      time.sleep(2)
44      matrix1.setBGcolor(0, 255, 255)         # 41
45      time.sleep(2)
46      matrix1.setBGcolor(255, 255, 255)        # 41
47      time.sleep(2)
48
49      matrix1.getMatrix().light(False)        # 43
50      time.sleep(3)
51
52      matrix1.finish()                      # 44
53      print('\nCase 1 square-Test erfolgreich.\n')
54      time.sleep(5)
```

Ergebnis Case 1: LEDmatrix('square')

- ✓ Test: Case 1 'square'
- ✓ Instanz LEDmatrix wurde erstellt.
- ✓ Verbindung zu brickd hergestellt.
- ✓ Callback aktiviert.
- ✓ square-index-Nr. 0, Name: square, Position: [[5, 10], [0, 10]]
- ✓ Press key to finish Thread
- ✓ Merge-Thread stopped
- ✓ Thread blink square stopped.
- ✓ Thread move square stopped.
- ✓ Press key to exit
- ✓ Verbindung beendet.
- ✓ Case 1 square-Test erfolgreich.

Case 2: LEDmatrix('colorHeartrate')

```
56     print("Test: Case 2 'colorHeartrate'\n")  # =====
57     matrix2 = LEDmatrix('colorHeartrate')
58     images2 = matrix2.getObjects()                      # 42
59
60     i = 0
61     while (i < len(images2)):
62         print('\n%s-index-Nr. %d, Name: %s, Position: ' % (matrix2.getCase(), i, images2[i].getName()),
63               images2[i].getMatrixPosition())             # 45, 39, 9
64         i += 1
65
66     time.sleep(2)
67
68     i = 0
69     while (i < len(images2)):
70         if (images2[i].getName() == 'bottomLeftImage'):    # 39
71             images2[i].setPuls(30)                         # 36
72
73         if (images2[i].getName() == 'bottom2ndLeftImage'): # 39
74             images2[i].setPuls(60)                         # 36
75
76         if (images2[i].getName() == 'bottom3rdLeftImage'): # 39
77             images2[i].setPuls(120)                        # 36
78
79         if (images2[i].getName() == 'bottom4thLeftImage'): # 39
80             images2[i].setPuls(180)                        # 36
81
82         i += 1
83     time.sleep(2)
84
85     time.sleep(2)
86
87     i = 0
88     while (i < len(images2)):
89         if (images2[i].getName() == 'bottomLeftImage'):    # 39
90             images2[i].setLightOffImageColor(0, 255, 255) # 15
91
92         if (images2[i].getName() == 'bottom2ndLeftImage'): # 39
93             images2[i].setLightOffImageColor(255, 0, 255) # 15
94
95         if (images2[i].getName() == 'bottom3rdLeftImage'): # 39
96             images2[i].setLightOffImageColor(0, 100, 0)   # 15
97
98         if (images2[i].getName() == 'bottom4thLeftImage'): # 39
99             images2[i].setLightOffImageColor(255, 255, 255) # 15
100
101        i += 1
102    time.sleep(2)
103
104    time.sleep(2)
105
106    matrix2.getMatrix().light(False)                      # 43
107    time.sleep(3)
108
109    matrix2.finish()                                     # 44
110    print('\nCase 2 colorHeartrate-Test erfolgreich.\n')
111    time.sleep(5)
```

Ergebnis Case 2: LEDmatrix('colorHeartrate')

- ✓ Test: Case 2 'colorHeartrate'
- ✓ Instanz LEDmatrix wurde erstellt.
- ✓ Verbindung zu brickd hergestellt.
- ✓ Callback aktiviert.
- ✓ colorHeartrate-index-Nr. 0, Name: topImage, Position: [[0, 20], [0, 5]]
- ✓ colorHeartrate-index-Nr. 1, Name: bottomLeftImage, Position: [[0, 5], [5, 5]]
- ✓ colorHeartrate-index-Nr. 2, Name: bottom2ndLeftImage, Position: [[5, 5], [5, 5]]
- ✓ colorHeartrate-index-Nr. 3, Name: bottom3rdLeftImage, Position: [[10, 5], [5, 5]]
- ✓ colorHeartrate-index-Nr. 4, Name: bottom4thLeftImage, Position: [[15, 5], [5, 5]]
- ✓ Press key to finish Thread
- ✓ Merge-Thread stopped
- ✓ Thread blink topImage stopped.
- ✓ Thread blink bottomLeftImage stopped.
- ✓ Thread blink bottom2ndLeftImage stopped.
- ✓ Thread blink bottom3rdLeftImage stopped.
- ✓ Thread blink bottom4thLeftImage stopped.
- ✓ Thread move topImage stopped.
- ✓ Thread move bottomLeftImage stopped.
- ✓ Thread move bottom2ndLeftImage stopped.
- ✓ Thread move bottom3rdLeftImage stopped.
- ✓ Thread move bottom4thLeftImage stopped.
- ✓ Press key to exit
- ✓ Verbindung beendet.
- ✓ Case 2 colorHeartrate-Test erfolgreich.

Case 3: LEDmatrix('quarters4')

```
113     print("\nTest: Case 3 'quarters4'\n") # =====
114     matrix3 = LEDmatrix('quarters4')
115     images3 = matrix3.getObjects()           # 42
116
117     i = 0
118     while (i < len(images3)):
119         print('\n%s-index-Nr. %d, Name: %s, Position: ' % (matrix3.getCase(), i, images3[i].getName()),
120               images3[i].getMatrixPosition())      # 45, 39, 9
121         i += 1
122
123     time.sleep(2)
124
125     matrix3.getMatrix().light(False)        # 43
126     time.sleep(3)
127
128     matrix3.finish()                      # 44
129     print('\nCase 3 quarters4-Test erfolgreich.\n')
130     time.sleep(3)
```

Ergebnis Case 3: LEDmatrix('quarters4')

- ✓ Test: Case 3 'quarters4'
- ✓ Instanz LEDmatrix wurde erstellt.
- ✓ Verbindung zu brickd hergestellt.
- ✓ Callback aktiviert.
- ✓ quarters4-index-Nr. 0, Name: topLeftImage, Position: [[0, 10], [0, 5]]
- ✓ quarters4-index-Nr. 1, Name: topRightImage, Position: [[10, 10], [0, 5]]
- ✓ quarters4-index-Nr. 2, Name: bottomLeftImage, Position: [[0, 10], [5, 5]]
- ✓ quarters4-index-Nr. 3, Name: bottomRightImage, Position: [[10, 10], [5, 5]]
- ✓ Press key to finish Thread
- ✓ Merge-Thread stopped
- ✓ Thread blink topLeftImage stopped.
- ✓ Thread blink topRightImage stopped.
- ✓ Thread blink bottomLeftImage stopped.
- ✓ Thread blink bottomRightImage stopped.
- ✓ Thread move topLeftImage stopped.
- ✓ Thread move topRightImage stopped.
- ✓ Thread move bottomLeftImage stopped.
- ✓ Thread move bottomRightImage stopped.
- ✓ Press key to exit
- ✓ Verbindung beendet.
- ✓ Case 3 quarters4-Test erfolgreich.

Case 4: LEDmatrix('row2')

```
132     print("\nTest: Case 4 'row2'\n")  # =====
133     matrix4 = LEDmatrix('row2')
134     images4 = matrix4.getObjects()           # 42
135
136     i = 0
137     while (i < len(images4)):
138         print('\n%s-index-Nr. %d, Name: %s, Position: ' % (matrix4.getCase(), i, images4[i].getName()),
139               images4[i].getMatrixPosition())      # 45, 39, 9
140         i += 1
141
142     time.sleep(2)
143
144     matrix4.getMatrix().light(False)          # 43
145     time.sleep(3)
146
147     matrix4.finish()                         # 44
148     print('\nCase 4 row2-Test erfolgreich.\n')
149     time.sleep(5)
```

Ergebnis Case 4: LEDmatrix('row2')

- ✓ Test: Case 4 'row2'
- ✓ Instanz LEDmatrix wurde erstellt.
- ✓ Verbindung zu brickd hergestellt.
- ✓ Callback aktiviert.
- ✓ row2-index-Nr. 0, Name: topImage, Position: [[0, 20], [0, 5]]
- ✓ row2-index-Nr. 1, Name: bottomImage, Position: [[0, 20], [5, 5]]
- ✓ Press key to finish Thread
- ✓ Merge-Thread stopped
- ✓ Thread blink topImage stopped.
- ✓ Thread blink bottomImage stopped.
- ✓ Thread move topImage stopped.
- ✓ Thread move bottomImage stopped.
- ✓ Press key to exit
- ✓ Verbindung beendet.
- ✓ Case 4 row2-Test erfolgreich.

Case 5: LEDmatrix('col2')

```
151      print("\nTest: Case 5 'col2'\n")  # =====
152      matrix5 = LEDmatrix('col2')
153      images5 = matrix5.getObjects()           # 42
154
155      i = 0
156      while (i < len(images5)):
157          print('\n%s-index-Nr. %d, Name: %s, Position: ' % (matrix5.getCase(), i, images5[i].getName()),
158                  images5[i].getMatrixPosition())      # 45, 39, 9
159          i += 1
160
161      time.sleep(2)
162
163      matrix5.getMatrix().light(False)        # 43
164      time.sleep(3)
165
166      matrix5.finish()                      # 44
167      print('\nCase 5 col2-Test erfolgreich.\n')
168      time.sleep(5)
```

Ergebnis Case 5: LEDmatrix('col2')

- ✓ Test: Case 5 'col2'
- ✓ Instanz LEDmatrix wurde erstellt.
- ✓ Verbindung zu brickd hergestellt.
- ✓ Callback aktiviert.
- ✓ col2-index-Nr. 0, Name: leftImage, Position: [[0, 10], [0, 10]]
- ✓ col2-index-Nr. 1, Name: rightImage, Position: [[10, 10], [0, 10]]
- ✓ Press key to finish Thread
- ✓ Merge-Thread stopped
- ✓ Thread blink leftImage stopped.
- ✓ Thread blink rightImage stopped.
- ✓ Thread move leftImage stopped.
- ✓ Thread move rightImage stopped.
- ✓ Press key to exit
- ✓ Verbindung beendet.
- ✓ Case 5 col2-Test erfolgreich.

Case 6: LEDmatrix('single')

```
170      print("\nTest: Case 6 'single'\n")  # =====
171      matrix6 = LEDmatrix('single')
172      images6 = matrix6.getObjects()           # 42
173
174      i = 0
175      while (i < len(images6)):
176          print('\ns-index-Nr. %d, Name: %s, Position: ' % (matrix6.getCase(), i, images6[i].getName()),
177                images6[i].getMatrixPosition())       # 45, 39, 9
178          i += 1
179
180      time.sleep(2)
181
182      matrix6.getMatrix().light(False)        # 43
183      time.sleep(3)
184
185      matrix6.finish()                      # 44
186      print('\nCase 6 single-Test erfolgreich.\n')
187      time.sleep(5)
```

Ergebnis Case 6: LEDmatrix('single')

- ✓ Test: Case 6 'single'
- ✓ Instanz LEDmatrix wurde erstellt.
- ✓ Verbindung zu brickd hergestellt.
- ✓ Callback aktiviert.
- ✓ single-index-Nr. 0, Name: singleImage, Position: [[0, 20], [0, 10]]
- ✓ Press key to finish Thread
- ✓ Merge-Thread stopped
- ✓ Thread blink singleImage stopped.
- ✓ Thread move singleImage stopped.
- ✓ Press key to exit
- ✓ Verbindung beendet.
- ✓ Case 6 single-Test erfolgreich.

Testfile: Test_split_merge_171110_images Letztmals erfolgreich: 01.12.2017

```

1  from LEDmatrix_split_merge_171110 import *
2  import imageProperties
3  import time
4
5
6  ▶ if __name__ == "__main__":
7
8      print("\nBildtests: Images von imageProperties \n") # -----
9      matrix = LEDmatrix()
10     images = matrix.getObjects()
11     imageObj = images[0]
12
13     print('defaultImage: Spalten (x) = %d | Zeilen (y) = %d\n' % (
14         len(imageProperties.defaultImage), len(imageProperties.defaultImage[0])))
15     imageObj.setInputImage(imageProperties.defaultImage)
16     time.sleep(2)
17
18     print(
19         'hello: Spalten (x) = %d | Zeilen (y) = %d\n' % (len(imageProperties.hello), len(imageProperties.hello[0])))
20     imageObj.setInputImage(imageProperties.hello)
21     time.sleep(2)
22
23     print(
24         'hearts1: Spalten (x) = %d | Zeilen (y) = %d\n' % (
25             len(imageProperties.hearts1), len(imageProperties.hearts1[0])))
26     imageObj.setInputImage(imageProperties.hearts1)
27     time.sleep(2)

```

- ✓ defaultImage: Spalten (x) = 20 | Zeilen (y) = 10
- ✓ hello: Spalten (x) = 20 | Zeilen (y) = 10
- ✓ hearts1: Spalten (x) = 20 | Zeilen (y) = 10
- ✓ forms1: Spalten (x) = 20 | Zeilen (y) = 10
- ✓ legoface: Spalten (x) = 20 | Zeilen (y) = 10
- ✓ hiOli: Spalten (x) = 42 | Zeilen (y) = 10
- ✓ subsets: Spalten (x) = 40 | Zeilen (y) = 10
- ✓ heartblue: Spalten (x) = 5 | Zeilen (y) = 5
- ✓ heartred: Spalten (x) = 5 | Zeilen (y) = 5
- ✓ heartgreen: Spalten (x) = 5 | Zeilen (y) = 5
- ✓ heartyellow: Spalten (x) = 5 | Zeilen (y) = 5
- ✓ pulstext: Spalten (x) = 20 | Zeilen (y) = 5
- ✓ hochformat: Spalten (x) = 20 | Zeilen (y) = 50
- ✓ querformat: Spalten (x) = 50 | Zeilen (y) = 10
- ✓ bild11Zeilen: Spalten (x) = 20 | Zeilen (y) = 11
- ✓ b11z: Spalten (x) = 20 | Zeilen (y) = 11
- ✓ whiteLine: Spalten (x) = 20 | Zeilen (y) = 11
 → Die weisse Linie ist beim Standbild nicht sichtbar, befindet sich auf der 11. Zeile. Nur bei moveUp() / moveDown()
- ✓ testImage: Spalten (x) = 10 | Zeilen (y) = 6

Testfile: Test_split_merge_171110_moves Letztmals erfolgreich: 01.12.2017

```
1  from LEDmatrix_split_merge_171110 import *
2  import imageProperties
3  import time
4
5  ▶ if __name__ == "__main__":
6
7
8      print("\nBewegungs-Test: Case 'square'\n")  # =====
9      matrix = LEDmatrix('square')
10     images = matrix.getObjects()
11     imageObj = images[0]
12
13     time.sleep(2)
14
15     # auch schön mit imageProperties.hearts1
16     imageObj.setInputImage(imageProperties.bild11Zeilen)
17     time.sleep(2)
18
19     imageObj.setMoveTime(0.1)
20     imageObj.setRate(10)
21
22     imageObj.moveRight()
23     time.sleep(2)
24
25     imageObj.startBlink()
26     time.sleep(2)
27     imageObj.stopBlink()
28     time.sleep(2)
29
30     imageObj.moveLeft()
31     time.sleep(2)
32     imageObj.stopMoveHorizontal()
33     time.sleep(2)
34
35     imageObj.moveUp()
36     time.sleep(2)
37     imageObj.moveDown()
38     time.sleep(2)
39     imageObj.stopMoveVertical()
40     time.sleep(2)
41
42     imageObj.moveRight()
43     imageObj.moveUp()
44     time.sleep(3)
45
46     imageObj.moveDown()
47     time.sleep(3)
48
49     imageObj.moveLeft()
50     time.sleep(3)
51
52     imageObj.moveUp()
53     time.sleep(3)
54
55     imageObj.resetMove()
56
57
58     matrix.getMatrix().light(False)
59     time.sleep(3)
60
61     matrix.finish()
62     print('\nBewegungs-Test erfolgreich.\n')
63     time.sleep(5)
```

Modul: imageProperties

Enthält Hauptsächlich eine Sammlung von Bildern (Listen von Farbinformationen), die als Input-Bilder genutzt werden können.

Information zum Auslesen der RGB-Farbinformationen aus dem Format [0, 0, 0]:

- IndexRed = 0
- IndexGreen = 1
- IndexBlue = 2

Praktische Funktionen:

Um den Bildarray auszugeben:

```
for x in defaultImage:  
    print(x)
```

Um die Anzahl Spalten und Zeilen eines Bildes auszugeben:

```
print('Spalten (x): %d | Zeilen (y): %d\n' % (len(defaultImage), len(defaultImage[0])))
```

Um ein Subset eines Bildes zu erstellen:

```
708     def getSubsetImage(image, imageFrom, imageTo):  
709         subsetImage = [[] for i in range(imageTo - (imageFrom - 1))]  
710         i = imageFrom - 1  
711         j = 0  
712         while i < imageTo:  
713             subsetImage[j] = image[i]  
714             i += 1  
715             j += 1  
716  
717         # subset = [subsets, subsetImage]  
718  
719     return subsetImage  
720  
721  
722 heartblue = getSubsetImage(hearts1, 2, 6)  
723 print(heartblue)
```

imageProperties-Bilder

- | | |
|-----------------|------------------------------------|
| ➤ defaultImage: | Spalten (x) = 20 Zeilen (y) = 10 |
| ➤ hello: | Spalten (x) = 20 Zeilen (y) = 10 |
| ➤ hearts1: | Spalten (x) = 20 Zeilen (y) = 10 |
| ➤ forms1: | Spalten (x) = 20 Zeilen (y) = 10 |
| ➤ legoface: | Spalten (x) = 20 Zeilen (y) = 10 |
| ➤ hiOli: | Spalten (x) = 42 Zeilen (y) = 10 |
| ➤ subsets: | Spalten (x) = 40 Zeilen (y) = 10 |
| ➤ heartblue: | Spalten (x) = 5 Zeilen (y) = 5 |
| ➤ heartred: | Spalten (x) = 5 Zeilen (y) = 5 |
| ➤ heartgreen: | Spalten (x) = 5 Zeilen (y) = 5 |
| ➤ heartyellow: | Spalten (x) = 5 Zeilen (y) = 5 |
| ➤ pulstext: | Spalten (x) = 20 Zeilen (y) = 5 |
| ➤ hochformat: | Spalten (x) = 20 Zeilen (y) = 50 |
| ➤ querformat: | Spalten (x) = 50 Zeilen (y) = 10 |
| ➤ bild11Zeilen: | Spalten (x) = 20 Zeilen (y) = 11 |
| ➤ b11z: | Spalten (x) = 20 Zeilen (y) = 11 |
| ➤ whiteLine: | Spalten (x) = 20 Zeilen (y) = 11 |
| ➤ testImage: | Spalten (x) = 10 Zeilen (y) = 6 |

Modul: matrixProperties

Enthält alle wichtigen Eigenschaftsinformationen des betriebenen LED-Displays, sowie Informationen, welche für das Funktionieren der *LEDmatrixConnect*-Klasse nötig sind.

Verbindungsinformationen:

- HOST = "localhost"
- PORT = 4223
- UIDbricklet = "wVj"
- UIDmaster = "6et15y"

Display-Dimensionen:

- ROWS = 10
- COLUMNS = 20
- NUM_LEDS = 16
- START_LED = COLUMNS - 1 # {19x} im Koordinatensystem
- LED_MAX = ROWS * COLUMNS - 1
- MATRIX_DEFAULT = [[0 for i in range(ROWS)] for i in range(COLUMNS)] # [[0 for i in range(10)] for i in range(20)]

Information zum Hineinlesen der RGB-Farbinformationen in das Format [0, 0, 0]:

- IndexRed = 0
- IndexGreen = 1
- IndexBlue = 2

defaultImage = [

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
[[0, 0, 0], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255],
[255, 255, 255], [255, 255, 255], [0, 0, 0]],
[[0, 0, 0], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255],
[255, 255, 255], [255, 255, 255], [0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [255, 255, 255], [255, 255, 255], [0, 0, 0], [0, 0, 0], [0, 0, 0],
[0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [255, 255, 255], [255, 255, 255], [0, 0, 0], [0, 0, 0], [0, 0, 0],
[0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [255, 255, 255], [255, 255, 255], [0, 0, 0], [0, 0, 0], [0, 0, 0],
[0, 0, 0]],
[[0, 0, 0], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255],
[255, 255, 255], [255, 255, 255], [0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [255, 255, 255], [255, 255, 255], [0, 0, 0], [0, 0, 0], [0, 0, 0],
[0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [255, 255, 255], [255, 255, 255], [0, 0, 0], [0, 0, 0], [0, 0, 0],
[0, 0, 0]],
[[0, 0, 0], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255], [255, 255, 255],
[255, 255, 255], [255, 255, 255], [0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0],
[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0],
[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0],
[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0],
[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0],
[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]]
```

Praktische Funktionen:

Um folgendes Array zu erstellen: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

```
def colorRow():
    return [0 for i in range(NUM_LEDS)] # [0 for i in range(16)]
```

Um folgendes Array zu bilden:

[[], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]

```
def colorRowsRGB():
    return [[0 for i in range(NUM_LEDS)] for i in range(3)] # [[0 for i in range(16)] for i in range(3)]
```

Um die Lämpchen des LED-Displays durchzunummerieren:

```
63     def LEDnr():
64         value = 0
65         array = MATRIX_DEFAULT # [[0 for i in range(10)] for i in range(20)]
66
67         x = START_LED # 19
68         while x > -1:
69             if x % 2 == 0: # von y0 nach yMax (=ROWS-1)
70                 y = 0
71                 while y < ROWS:
72                     array[x][y] = value
73                     y += 1
74                     value += 1
75             else: # von yMax (=ROWS-1) nach y0
76                 y = ROWS - 1
77                 while y > -1:
78                     array[x][y] = value
79                     y -= 1
80                     value += 1
81
82             x -= 1
83
84
85     LEDnr = LEDnr()
86
```

Um die Lämpchen-Nummer aus der Liste herauszulesen (Liste von LEDnr())

```
def getLEDnr(xMatrix, yMatrix):
    return LEDnr[xMatrix][yMatrix]
```

Um die Liste der Lämpchen-Nummern zu drucken (Liste von LEDnr())

```
def printLEDnr():
    for x in LEDnr:
        print(x)

printLEDnr()
```

Anhang A: Konsolen-Output der Testergebnisse von Test_split_merge_171110.py

Ergebnis Case 0: LEDmatrix()

```
Test: Case 0 'default'

Instanz LEDmatrix wurde erstellt.
Verbindung zu brickd hergestellt.
Callback aktiviert.

Default-index-Nr. 0, Name: singleImage, Position: [[0, 20], [0, 10]]

Press key to finish Thread

Thread blink singleImage stopped.
Merge-Thread stopped

Thread move singleImage stopped.

Press key to exit

Verbindung beendet.

Case 0 Default(=single)-Test erfolgreich.
```

Ergebnis Case 1: LEDmatrix('square')

```
Test: Case 1 'square'

Instanz LEDmatrix wurde erstellt.
Verbindung zu brickd hergestellt.
Callback aktiviert.

square-index-Nr. 0, Name: square, Position: [[5, 10], [0, 10]]

Press key to finish Thread

Merge-Thread stopped
Thread blink square stopped.

Thread move square stopped.

Press key to exit

Verbindung beendet.

Case 1 square-Test erfolgreich.
```

Ergebnis Case 2: LEDmatrix('colorHeartrate')

```
Test: Case 2 'colorHeartrate'

Instanz LEDmatrix wurde erstellt.
Verbindung zu brickd hergestellt.
Callback aktiviert.

colorHeartrate-index-Nr. 0, Name: topImage, Position: [[0, 20], [0, 5]]
colorHeartrate-index-Nr. 1, Name: bottomLeftImage, Position: [[0, 5], [5, 5]]
colorHeartrate-index-Nr. 2, Name: bottom2ndLeftImage, Position: [[5, 5], [5, 5]]
colorHeartrate-index-Nr. 3, Name: bottom3rdLeftImage, Position: [[10, 5], [5, 5]]
colorHeartrate-index-Nr. 4, Name: bottom4thLeftImage, Position: [[15, 5], [5, 5]]

Press key to finish Thread

Thread blink topImage stopped.
Merge-Thread stopped

Thread blink bottom3rdLeftImage stopped.
Thread move topImage stopped.

Thread move bottom2ndLeftImage stopped.
Thread move bottomLeftImage stopped.

Thread blink bottom4thLeftImage stopped.
Thread move bottom3rdLeftImage stopped.
Thread move bottom4thLeftImage stopped.

Thread blink bottom2ndLeftImage stopped.
Thread blink bottomLeftImage stopped.

Press key to exit

Verbindung beendet.

Case 2 colorHeartrate-Test erfolgreich.
```

Ergebnis Case 3: LEDmatrix('quarters4')

```
Test: Case 3 'quarters4'

Instanz LEDmatrix wurde erstellt.
Verbindung zu brickd hergestellt.
Callback aktiviert.

quarters4-index-Nr. 0, Name: topLeftImage, Position: [[0, 10], [0, 5]]
quarters4-index-Nr. 1, Name: topRightImage, Position: [[10, 10], [0, 5]]
quarters4-index-Nr. 2, Name: bottomLeftImage, Position: [[0, 10], [5, 5]]
quarters4-index-Nr. 3, Name: bottomRightImage, Position: [[10, 10], [5, 5]]

Press key to finish Thread

Thread blink topLeftImage stopped.
Thread blink bottomRightImage stopped.
Thread blink topRightImage stopped.
Thread blink bottomLeftImage stopped.
Merge-Thread stopped

Thread move topRightImage stopped.
Thread move topLeftImage stopped.

Thread move bottomLeftImage stopped.

Thread move bottomRightImage stopped.

Press key to exit

Verbindung beendet.

Case 3 quarters4-Test erfolgreich.
```

Ergebnis Case 4: LEDmatrix('row2')

```
Test: Case 4 'row2'

Instanz LEDmatrix wurde erstellt.
Verbindung zu brickd hergestellt.
Callback aktiviert.

row2-index-Nr. 0, Name: topImage, Position: [[0, 20], [0, 5]]
row2-index-Nr. 1, Name: bottomImage, Position: [[0, 20], [5, 5]]
Press key to finish Thread

Thread blink bottomImage stopped.
Thread blink topImage stopped.
Merge-Thread stopped

Thread move bottomImage stopped.
Thread move topImage stopped.

Press key to exit
Verbindung beendet.

Case 4 row2-Test erfolgreich.
```

Ergebnis Case 5: LEDmatrix('col2')

```
Test: Case 5 'col2'

Instanz LEDmatrix wurde erstellt.
Verbindung zu brickd hergestellt.
Callback aktiviert.

col2-index-Nr. 0, Name: leftImage, Position: [[0, 10], [0, 10]]
col2-index-Nr. 1, Name: rightImage, Position: [[10, 10], [0, 10]]

Press key to finish Thread

Thread blink rightImage stopped.
Merge-Thread stopped
Thread blink leftImage stopped.

Thread move leftImage stopped.

Thread move rightImage stopped.

Press key to exit

Verbindung beendet.

Case 5 col2-Test erfolgreich.
```

Ergebnis Case 6: LEDmatrix('single')

```
Test: Case 6 'single'

Instanz LEDmatrix wurde erstellt.
Verbindung zu brickd hergestellt.
Callback aktiviert.

single-index-Nr. 0, Name: singleImage, Position: [[0, 20], [0, 10]]

Press key to finish Thread

Thread blink singleImage stopped.
Merge-Thread stopped

Thread move singleImage stopped.

Press key to exit

Verbindung beendet.

Case 6 single-Test erfolgreich.
```

Anhang B: Informations-Suchergebnisse

Python

- <https://py-tutorial-de.readthedocs.io/de/python-3.3/index.html>
- <http://www.physik.uzh.ch/local/teaching/PHY114/python/python-listen.php>
- http://www.python-kurs.eu/python3_kurs.php

Flask

Json

- <http://json.parser.online.fr/>

REST-API

- <https://poe-php.de/tutorial/rest-einfuehrung-in-die-api-erstellung>
- https://apigee.com/console/twitter?apig_cc=1
- <https://identity.getpostman.com/login>
- <https://www.youtube.com/watch?v=OTWs0wPHU-g>
- <https://www.youtube.com/watch?v=WxGBoY5iNXY>
- <https://www.youtube.com/watch?v=qVTAB8Z2VmA>
- <https://www.youtube.com/watch?v=7YcW25PHnAA>
- <https://www.youtube.com/watch?v=AQboSORESt4>
- <https://github.com/scotch-io/python-flask-crud-1>
- <https://scotch.io/tutorials/build-a-crud-web-app-with-python-and-flask-part-one>
- <http://python-eve.org/>
- <https://de.slideshare.net/larrycai/learn-rest-apiwithpython>
- <https://gist.github.com/larrycai/7823499>
- <http://www.youtube.com/watch?v=hdSrT4yjS1g>

Tkinter

Tinkerforge - Blinkenlights

- https://www.tinkerforge.com/de/doc/Hardware/Bricks/RED_Brick.html
- https://www.tinkerforge.com/de/doc/Software/Bricks/RED_Brick_Python.html#red-brick-python-examples
- https://www.tinkerforge.com/de/doc/Software/Bricks/RED_Brick_Python.html#red-brick-python-api
- <https://www.tinkerforge.com/de/>
- https://www.tinkerforge.com/de/doc/Software/API_Bindings_Python.html
- <https://www.tinkerforge.com/de/doc/Tutorials/Tutorial.Authentication/Tutorial.html#mit-callbacks>

Bluetooth 4.0

Gatt-Spezifikation:

- <https://www.bluetooth.com/specifications/gatt>
- https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.characteristic.heart_rate_measurement.xml
- <https://www.bluetooth.com/develop-with-bluetooth/developer-resources-tools/developer-kits>
- <https://www.bluetooth.com/develop-with-bluetooth/service-providers/software-application-development>
- <https://www.bluetooth.com/develop-with-bluetooth/test-tools>
- https://epxx.co/artigos/bluetooth_gatt.html
- <https://www.bluetooth.com/specifications/assigned-numbers/units>
- https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.characteristic.heart_rate_measurement.xml
- <https://www.bluetooth.com/specifications/gatt/characteristics>

- https://www.u-blox.com/sites/default/files/products/documents/BluetoothLowEnergyGATT-Examples_AppNote_%28cB-23035914%29.pdf
- http://api.wahoofitness.com/android/beta/classcom_1_1wahoofitness_1_1api_1_1w_f.hardware_connector_types_1_1w_f.sensor_type.html

- <https://stackoverflow.com/questions/46191704/retrieve-calories-from-ble-devices-eg-mio-fuse>
- <https://stackoverflow.com/questions/44374505/utilising-bluetooth-on-mac-with-python>
- <https://bitbucket.org/OscarAcena/pygattlib/src/45e04060881a20189412681f52d55ff5add9f388/DEPENDS?at=default>

- <http://www.didgy.org.uk/tracker-app-adding-heart-rar/>
- <https://evothings.com/forum/viewtopic.php?t=1530>

- <https://github.com/evothings/evothings-examples/blob/master/examples/mbed-custom-gatt/app.js>
- <https://github.com/evothings/evothings-examples/tree/master/examples>
- <https://github.com/evothings/evothings-examples/blob/master/examples/ble-scan/app.js>
- <https://github.com/sandeepmistry/noble>

- https://github.com/adafruit/Adafruit_Python_BluefruitLE
- <https://learn.adafruit.com/assets/27226>
- <https://github.com/2bbb/ofxMioAlpha>
- <https://github.com/2bbb/ofxMioAlpha>

- <https://pythonhosted.org/pyobjc/install.html#manual-installation>

- <https://github.com/karulis/pybluez>

- <http://lightblue.sourceforge.net/doc/index.html>
- <http://lightblue.sourceforge.net/LightAquaBlue/index.html>
- <http://lightblue.sourceforge.net/>
- <https://pypi.python.org/pypi/lightblue/0.2>

- <https://developer.apple.com/bluetooth/>
- https://developer.apple.com/library/content/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/PerformingCommonPeripheralRoleTasks/PerformingCommonPeripheralRoleTasks.html
- http://www.mt-system.ru/sites/default/files/documents/ble_application_note_hrp_sensor.pdf
- https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.characteristic.heart_rate_measurement.xml&u=org.bluetooth.characteristic.heart_rate_measurement.xml
- <https://stackoverflow.com/questions/20607074/how-to-identify-uuid-of-ble-from-bluetooth-service-specifications>
- http://dev.ti.com/tirex/content/simplelink_cc2640r2_sdk_1_35_00_33/docs/ble5stack/ble_user_guide/html/ble-stack/gatt.html

- <http://support.connectblue.com/pages/viewpage.action?pageId=23035914>

- <https://stackoverflow.com/questions/10421633/analyze-data-return-from-heart-rate-monitor>

- https://github.com/search?l=Python&langOverride=&q=CoreBluetooth&repo=&start_value=1&type=Repositories&utf8=%E2%9C%93

- https://developer.apple.com/library/content/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth_Introduction.html#/apple_ref/doc/uid/TP40013257
- <https://developer.apple.com/documentation/corebluetooth>
- https://developer.apple.com/library/content/releasenotes/General/APIDiffsMacOS10_12/Swift/CoreBluetooth.html
- <https://developer.apple.com/search/?q=CoreBluetooth>
- <https://stackoverflow.com/questions/12754649/bluetooth-4-bluetooth-smart-on-mac-os>

- <https://itunes.apple.com/us/app/lightblue/id639944780?mt=12>
- <https://itunes.apple.com/de/app/bluetooth-smart-scanner/id509978131?mt=8>
- <http://www.itwissen.info/Bluetooth-4DOT-0-Bluetooth-smart.html>
- <http://de.ccm.net/contents/605-funktionsweise-von-bluetooth>

Versionskontrolle

Dokument befindet sich unter:

➤ smb://bfh.ch/data/ti/medinf/MedInf_Team/Labor/LEDmatrix_Blinkenlights_Tinkerforge

Code auf github: https://github.com/caldfl/BFH_MatrixLED.git