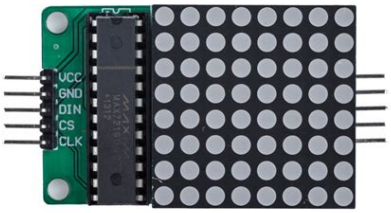
Pong mit einer LED-Matrix

Aufgabe: Wir wollen lernen wie man externe Erweiterungen mit dem Raspberry Pi verwendet. In diesem Fall ist es eine LED-Matrix. Um das ganze spannend zu gestalten werden wir das machen indem wir ein Spiel dafür programmieren. Pong bietet sich hier an, da wir keine hohe Auflösung haben, sondern nur eine 8x8 Matrix. Das heißt wir können insgesamt 64 Pixel darstellen.
Die Aufgabe ist also Pong zu programmieren und es auf der Matrix anzuzeigen.

Die LED-Matrix:



Die LED-Matrix ist ein Block mit 8x8 LED's . Auf der einen Seite ist noch ein Block namens „Max2719".

Was macht der Max2719 denn eigentlich?
Um alle LED's der Matrix einzelne ansteuern zu können bräuchte es eigentlich 64 eigene Anschlüsse. Der Max2719 Baustein arbeitet mit der Stromstärke und simuliert mit verschiedenen Stromstärken mehrere Anschlüsse. Wie genau das funktioniert ist aber nicht wichtig.

Anschliessen der LED-Matrix:

Board Pin	Name	Remarks	RPi Pin	RPi Function
1	VCC	+5V Power	2	5V0
2	GND	Ground	6	GND
3	DIN	Data In	19	GPIO 10 (MOSI)
4	CS	Chip Select	24	GPIO 8 (SPI CE0)
5	CLK	Clock	23	GPIO 11 (SPI CLK)

Auf der einen Seite sind verschiedene Begriffe notiert wie „VCC", „GND", „DIN", „CS" und „CLK". Für diese könnt ihr in der Tabelle nachgucken wo ihr sie anschließen könnt.
VCC wird z.B. an RPi Pin 2 angeschlossen.

Malen auf der LED-Matrix:

Um auf der LED-Matrix zu arbeiten verwenden wir zwei externe Libraries:

RaspberryPi Max27197 Treiber (Luma LED) - https://max7219.readthedocs.io/en/latest/#
- https://github.com/rm-hull/luma.led_matrix

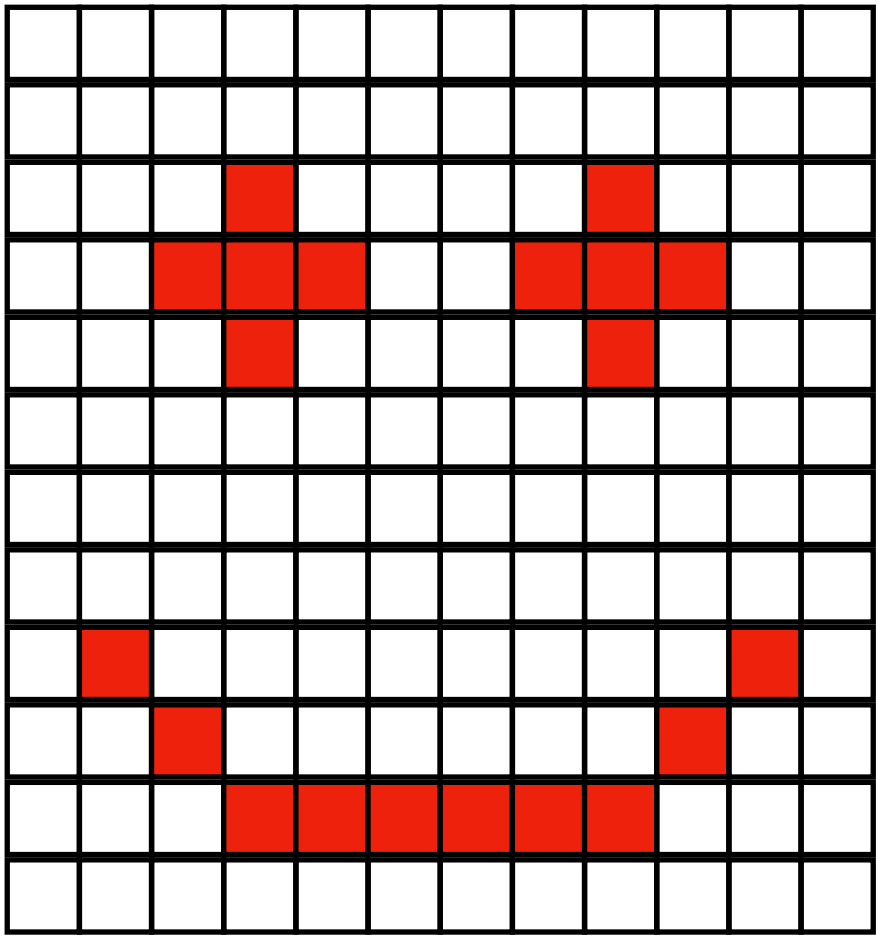
Pillow - https://pillow.readthedocs.io/en/stable/

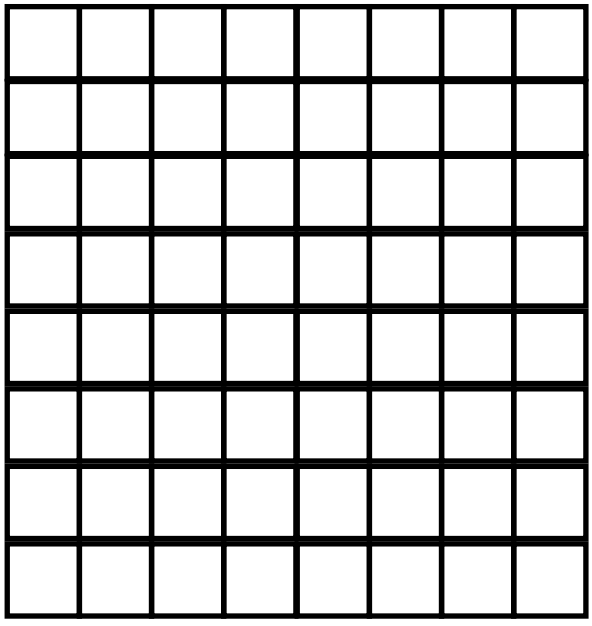
Pong mit einer LED-Matrix

Wie funktioniert das Anzeigen auf der Matrix mit Hilfe von Pillow:

Zuerst erstellen wir ein virtuelles Bild. Dieses kann eine beliebige Auflösung haben. Zum Beispiel nehmen wir 12x12.

Hier haben wir das virtuelle Bild mit einem Smiley bemalt.



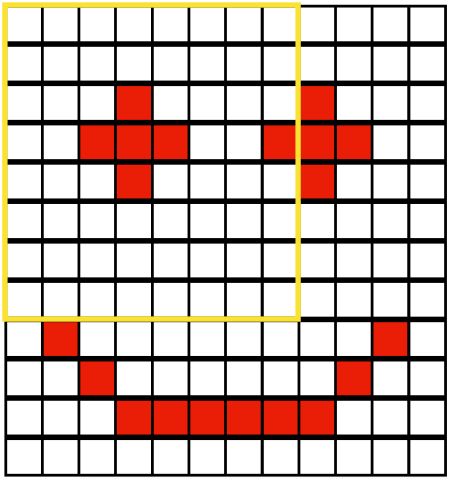


Neben dem virtuellen Bild haben wir auch die eigentliche Matrix und die hat wie gesagt nur 8x8.

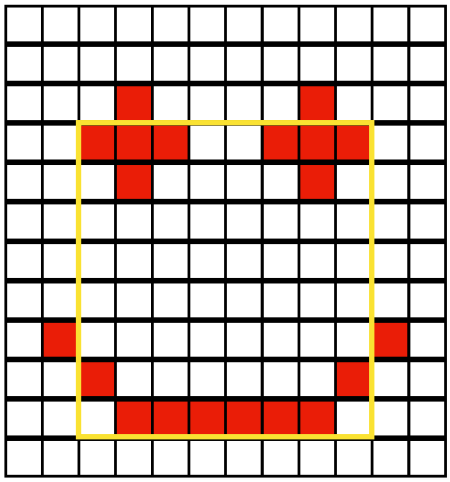
Pong mit einer LED-Matrix

Um jetzt in der Matrix etwas anzuzeigen müssen wir das virtuelle Bild über das von der Matrix. Das machen wir mit.

„virtual.set_position((variable1,variable2))“

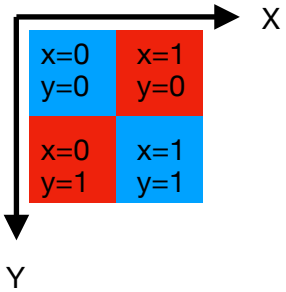


Tippen wir „virtual.set_position((0,0))“ so wird nur der Bereich in dem gelben Rahmen angezeigt (auf der Matrix).



Tippen wir aber nun „virtual.set_position((3,4))“ dann wird der Rahmen wie folgt gesetzt.

Allgemein gilt :



Die LED’s werden auf der x Achse von 0 bis … gezählt.
Die LED’s werden auf der y Achse von 0 bis … gezählt.

Pong mit einer LED-Matrix

Rahmenprogramm und seine Benutzung:
Für diese Aufgabe habt ihr wieder ein Rahmenprogramm gegeben, welches für euch mit der Matrix interagiert.

Dokumentation von Pillow and Luma:

```
player1 = [(0,3),(0,4)]
player2 = [(7,3),(7,4)]
dot = [(3,3),(4,4)]
```

Hier haben wir die Variablen für unsere beiden Spieler und die für unseren Punkt.
Sowohl die Spieler als auch der Punkt werden durch zwei Punkte dargestellt. Dies passiert als ein Array von Tupeln, aber erstmal klären wir was ein Array und was ein Tupel ist.

Array:

Ein Array solltet ihr bereits gehabt haben. Es ist im Prinzip eine Liste mit der Struktur **array = [x,y,z]**. In der Variable array liegen jetzt also unsere Variablen **x, y und z**. Um auf die zugreifen zu können schreiben wir:
Wert1 = array[0] # Wert1 ist jetzt x
Wert2 = array[1] # Wert2 ist jetzt y
Wert3 = array[2] # Wert3 ist jetzt z

Wichtig ist, dass ein Array bei 0 anfängt zu zählen!

Tupel:

Ein Tupel ist ähnlich zu einem Array. Der Unterschied ist, dass in einem Tupelo mehrere verschiedene Werte haben kann. Ihr solltet aber allgemein darauf achten keine zu großen Tupel zu verwenden. Tutel haben solch eine Struktur **tupel = (x,y)** mit den Variablen x und y. Um darauf zugreifen zu können schreiben wir:
Wert1 = tupel[0]
Wert2 = tupel[1]

Array aus Tupeln:

Haben wir ein Struktur wie diese **player1 = [(0,3),(0,4)]** dann gilt:

Wert1 = player1[0][0] # Wert1 ist jetzt 0
Wert2 = player1[1][1] # Wert 2 ist jetzt 4
Wert3 = player1[0][1] # Wert 3 ist jetzt 3
Wert4 = player1[1][0] # Wert 4 ist jetzt 0

Malen:

Ein Tupel steht hier für einen Punkt.
(x,y) Das x ist der Punkt auf der x-Koordinate und das y ist der Punkt auf der y-Koordinate.

Die Spieler ziehen eine Linie zwischen beiden Punkten und der Dot ein Rechteck zwischen beiden angegebenen Punkten.

(Probiert selber mal Werte für die Spieler und den Punkt aus um etwas mehr Verständnis zu bekommen.)