A pixelated, low-resolution portrait of a man. He is wearing a grey knit beanie with a dark stripe and a bright red jacket. He is looking out of a window, with a blurred city street and buildings visible in the background. The image has a distinct dithered or pixelated texture.

Portrait Machine

++Jongwon Choi
//mac1024@gmail.com
--www.khm.de/~mac1024

_ Description

'Portrait Machine' ist eine Lochmaschine, die die binären Daten von Profilbildern aus Facebook auf ein Papier abbildet.

Facebook erreichte im September 2011 rund 800 Millionen Mitglieder weltweit. Viele von ihnen benutzen Portraits als Profilbild. Die Facebook-Plattform bietet eine Programmierschnittstelle (API) über die eigene Anwendungen darauf zugreifen können.

Meine Anwendung greift auf die Bilddaten von Facebook-Profilen zu. Die binär codierten Daten, die den Bildern zu Grunde liegen, werden nach Farbinformationen sortiert (nach ihren Rot-, Grün, Blau-, und Transparenzanteilen) und an einen Mikrocontroller gesendet. Der Mikrocontroller steuert an Hand dieser Daten das Verhalten der 'Portrait Maschine'.

Die 'Portrait Machine' stellt so aus den Profil-Bilddaten ein neues Bild her.

_ Binary Code

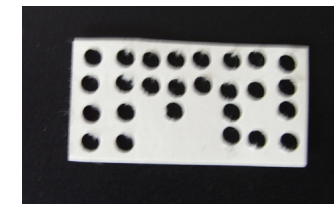
Binärcode ist die Gesamtheit aller Codes welche Informationen durch Sequenzen von zwei verschiedenen Symbolen (zum Beispiel 1/0 oder wahr/falsch) dargestellt werden können. Die Bezeichnung leitet sich von der lateinischen Vorsilbe bi ab, welche die Bedeutung zwei oder doppelt hat. Binär-codes bilden die Grundlage für jegliche Verarbeitung von digitaler Informationen. Sie lassen sich technisch sehr leicht abbilden und verarbeiten, z.B. durch Spannungen: Spannung liegt an □ entspricht 1 oder logisch wahr, Spannung liegt nicht an □ entspricht 0 oder logisch falsch.

_ Farben Informationen

Aus technischer Sicht, Farben 32 Bit an Informationen, wie AAAAAAAAAARRRRRRRRGGGGGGGG-
BBBBBBBBB bestellt, wo die A enthalten, der Alpha-Wert sind die R ist der Rot-Wert sind, G ist grün, und B ist blau. Jede Komponente ist 8 Bit (eine Zahl zwischen 0 und 255). Diese Werte können mit Bit-Verschiebung manipuliert werden.



A	R	G	B
11111111	11111111	11010101	11000111
(255	255	213	199)



Binary code

Lochkarte

_ Facebook API

The screenshot shows the Facebook Developers website. The top navigation bar includes 'facebook DEVELOPERS', 'Documentation', 'Support', 'Blog', 'Apps', and a search box. The left sidebar contains a menu with 'Getting Started', 'Core Concepts' (expanded to show 'Social Design', 'Social Plugins', 'Open Graph protocol', 'Social Channels', 'Authentication', and 'Graph API'), 'Advanced Topics', and 'SDKs & Tools'. The main content area is titled 'Graph API' with a sub-header 'Core Concepts > Graph API'. The text explains that Facebook's core is the social graph and that the Graph API provides a consistent view of it. A red highlighted box contains the text: 'Every object in the social graph has a unique ID. You can access the properties of an object by requesting <https://graph.facebook.com/ID>. For example, the official page for the Facebook Platform has id 19292868552, so you can fetch the object at <https://graph.facebook.com/19292868552>:' Below this, a code block shows a JSON object starting with '{

Every object in the social graph has a unique ID. You can access the properties of an object by requesting <https://graph.facebook.com/ID>. For example, the official page for the Facebook Platform has id 19292868552, so you can fetch the object at <https://graph.facebook.com/19292868552>:

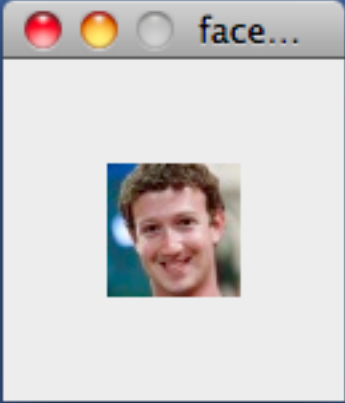
The screenshot shows a list of examples for Graph API endpoints. The left sidebar lists various object types like 'Event', 'FriendList', 'Group', etc. The main content area lists several examples, with one highlighted in red: 'Profile pictures: <https://graph.facebook.com/km1004/picture> (your profile picture)'. Other examples include Events, Groups, Applications, Status messages, Photos, Photo albums, Videos, Notes, and Checkins. At the bottom, it states: 'All of the objects in the Facebook social graph are connected to each other via relationships. Bret Taylor is a fan of the

Profile pictures: <https://graph.facebook.com/km1004/picture> (your profile picture)

ID

_ API programming

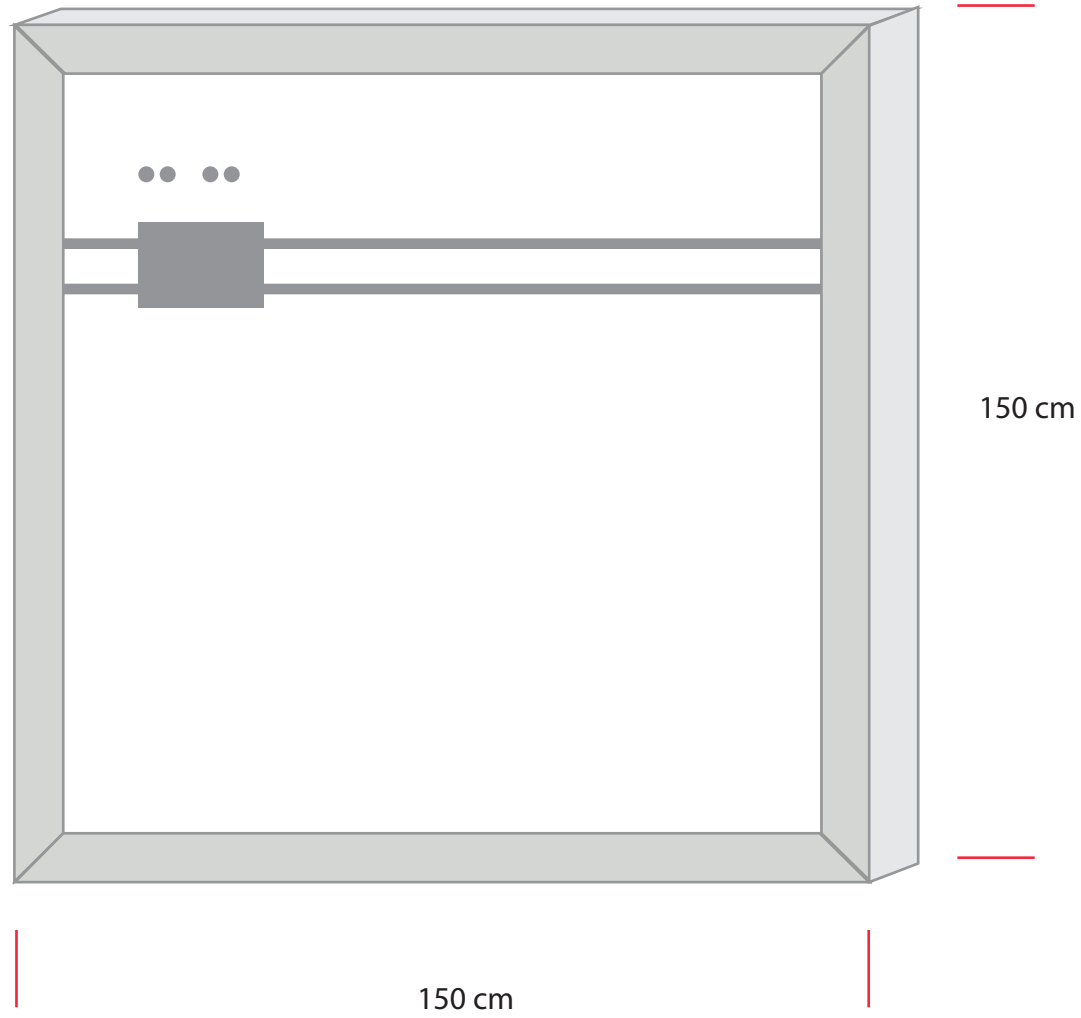
```
if ( i > 0);  
{  
    profilephoto = "https://graph.facebook.com/" + i + "/picture";  
    println (profilephoto);  
}  
  
profile = loadImage(profilephoto, "jpg");  
image(profile, 0, 0);  
i
```



```
https://graph.facebook.com/1/picture  
https://graph.facebook.com/2/picture  
https://graph.facebook.com/3/picture  
https://graph.facebook.com/4/picture  
https://graph.facebook.com/5/picture
```

23

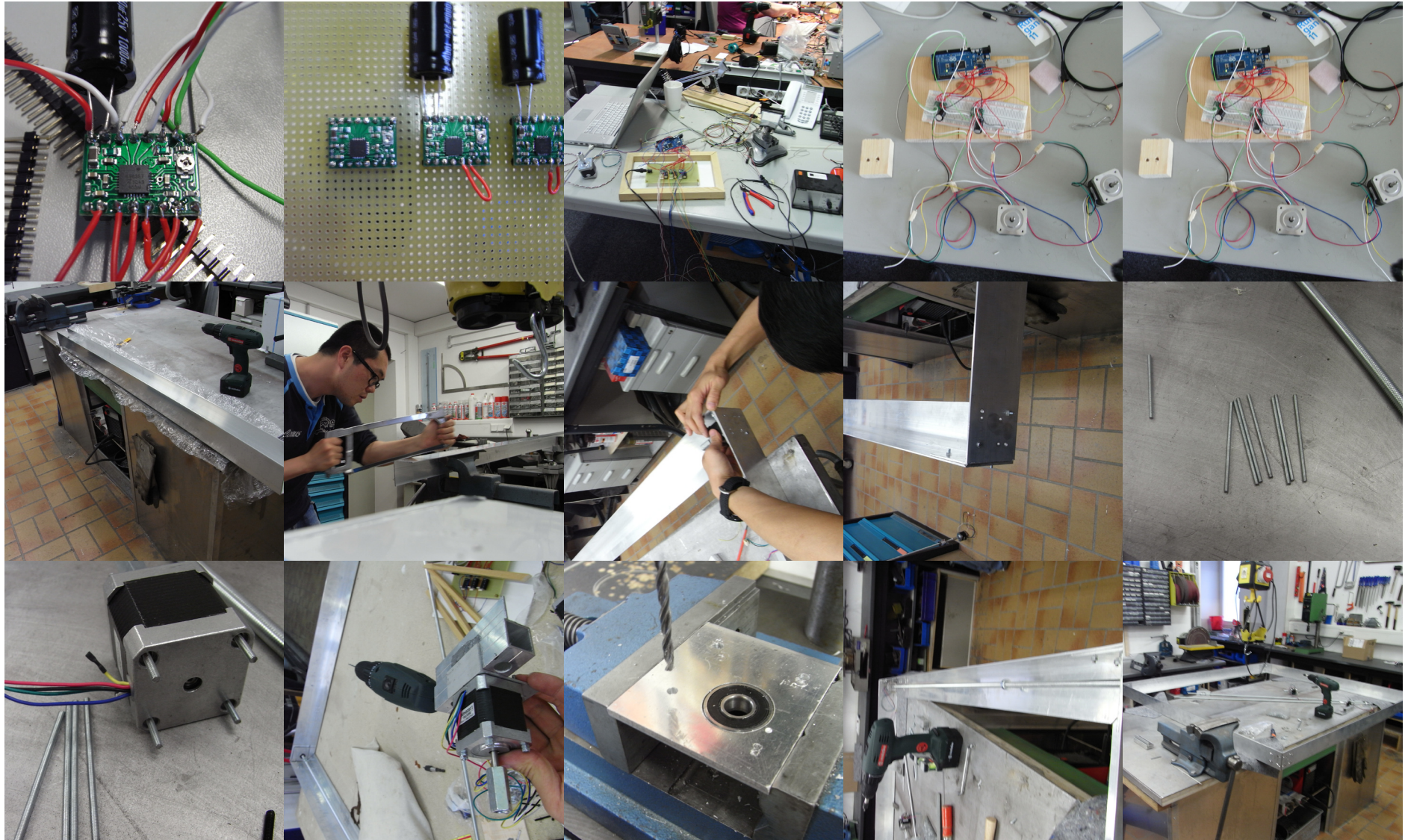
_ Portrait Machine



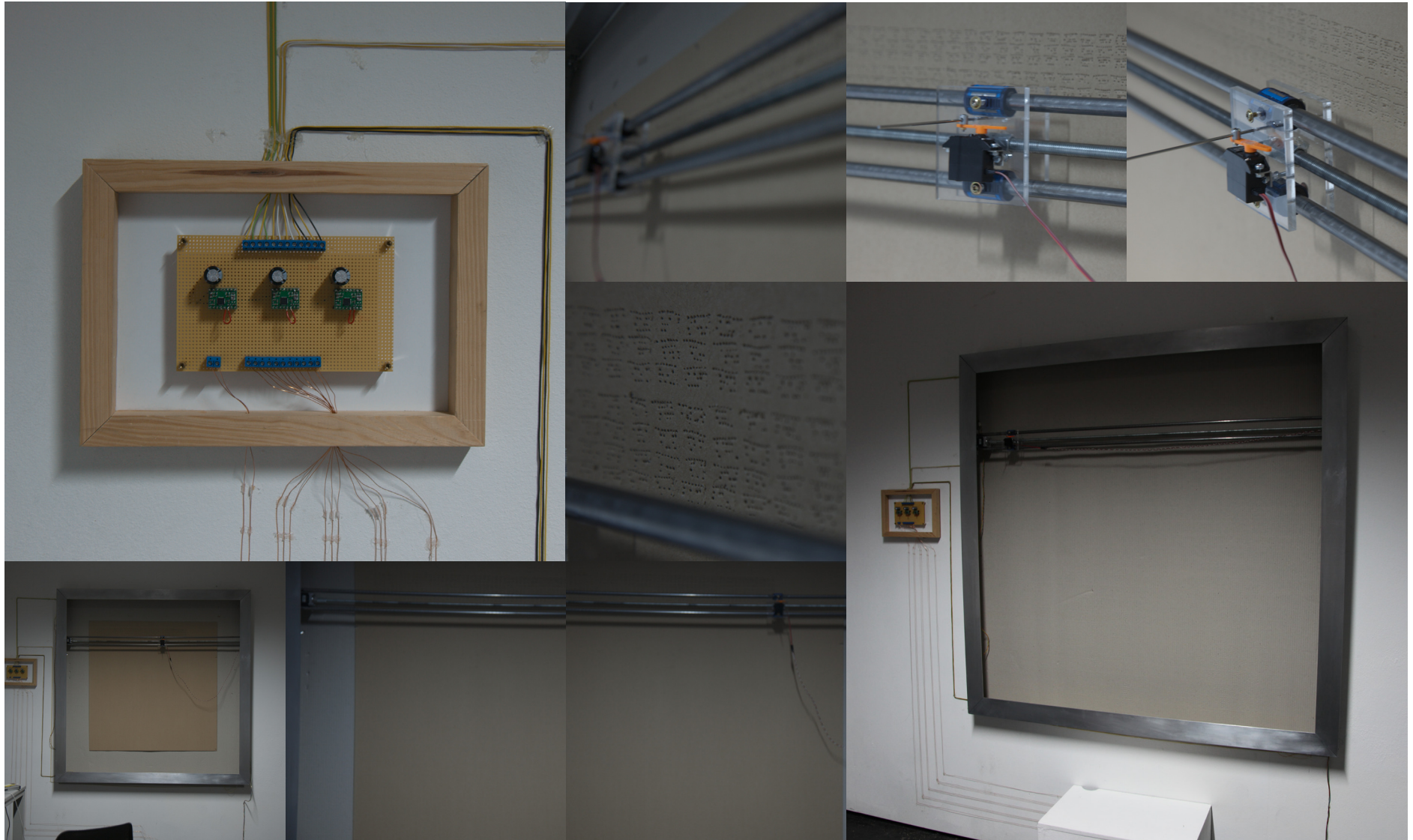
_ Portrait Machine



_ Development



_ Test Assembly



_ Printing

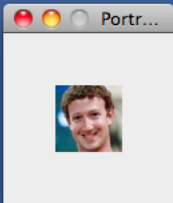


_ Printing

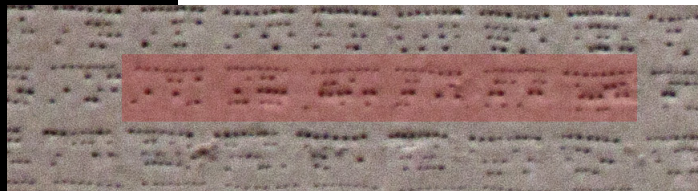


_ Portrait Bild Printing

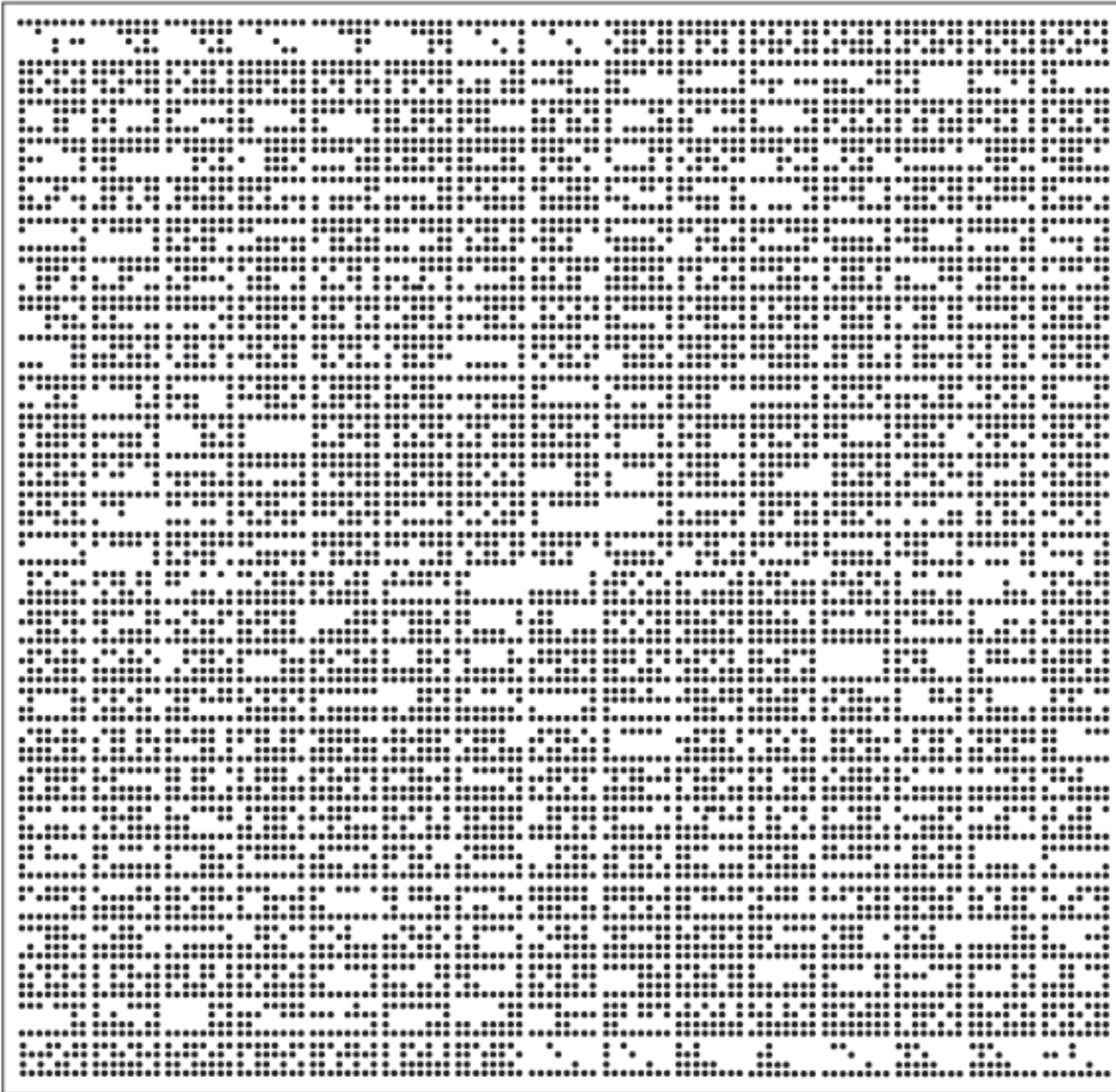
```
PortraitMachine §  
//output.println(binary(a) + '\t' + binary(r) + '\t' + binary(g) + '\t' +  
//output.println(binary(p));  
  
String out = binary(p);  
println (out);  
  
Thread.currentThread().sleep(pause);  
String a = out.substring(0,8)+"\n";  
port.write('A');  
for (int j=0; j<a.length(); j++) {  
    print(a.charAt(j));  
  
    port.write(a.charAt(j));  
    output.print(a.charAt(j));  
    Thread.currentThread().sleep(pause);  
}  
// R  
  
Thread.currentThread().sleep(pause);  
String r = out.substring(8,16)+"\n";  
port.write('R');  
for (int j=0; j<r.length(); j++) {  
    print(r.charAt(j));  
  
    port.write(r.charAt(j));  
    output.print(r.charAt(j));  
    Thread.currentThread().sleep(pause);  
}  
// G  
  
Thread.currentThread().sleep(pause);  
String g = out.substring(16,24)+"\n";  
port.write('G');  
for (int j=0; j<g.length(); j++) {  
    print(g.charAt(j));  
  
    port.write(g.charAt(j));  
    output.print(g.charAt(j));  
    Thread.currentThread().sleep(pause);  
}  
// B  
  
Thread.currentThread().sleep(pause);  
String b = out.substring(24,32)+"\n";  
port.write('B');  
for (int j=0; j<b.length(); j++) {  
    print(b.charAt(j));  
  
    port.write(b.charAt(j));  
    output.print(b.charAt(j));  
    Thread.currentThread().sleep(pause);  
}  
// P  
  
Thread.currentThread().sleep(pause);  
String p = out.substring(32,64)+"\n";  
port.write('P');  
for (int j=0; j<p.length(); j++) {  
    print(p.charAt(j));  
  
    port.write(p.charAt(j));  
    output.print(p.charAt(j));  
    Thread.currentThread().sleep(pause);  
}
```



```
11111111000110100100100010000011  
11111111  
00011010  
01001000  
10000011  
pixel 108  
11111111001011000100111101110111  
11111111  
00101100  
01001111  
01110111  
pixel 109  
11111111011010000111101110001001  
11111111  
01101000  
01111011  
10001001  
pixel 110  
11111111011000110110010101100000  
11111111  
01100011  
01100101  
01100000  
pixel 111  
11111111011100110110001101010100  
11111111  
.....  
26
```



_ Portrait Bild
Visualisierung



_ Portrait Bild

