

Alchemy – Eine explorative Playful Interaction



Team

Laurin Muth, Christoph Härtl

Konzept

Das Projekt *Alchemy* ist eine Tangible Playful Interaction. Unser Ziel bei der Entwicklung war es, ein leicht zu transportierendes System zu entwickeln, um eine „verspielte“, anfassbare Repräsentation des Themas der mittelalterlichen Alchemie zu schaffen.

Implementierung

Der Prototyp besteht aus einer Basis aus Holz, auf dem eine dreieckige Box steht, in der sich die Elektronik verbirgt, einem Rundkolben, in welchem sich mehrere hintereinander geschaltene LED-Streifen befinden, und einem selbstgebautem Reagenzglasständer, in welchem fünf Küvetten stehen in denen Repräsentationen von verfügbaren Elementen gefüllt sind, wie etwa blau gefärbtes Wasser oder Blätter. Auf dem Rundkolben befindet sich eine RFID-Antenne, auf den Deckeln der Küvetten RFID-Tags. So werden die Küvetten erkannt.

Status, Erweiterungsmöglichkeiten

Der Prototyp legt die grundlegenden Funktionen eines möglichen Endprodukts dar. Von besonderer Bedeutung sind dabei die bereits zufriedenstellend erfüllten Anforderungen einfach verständlicher Affordances sowie einem modularen Aufbau des Systems. Das System könnte noch um einen MP3-Player erweitert werden, um dem Nutzer zusätzliches Feedback zu geben. Auch denkbar wären weitere Interaktionsmöglichkeiten, wie beispielsweise eine „Rührfunktion“ am Kolben. Aufgrund der Position und Empfindlichkeit der RFID-Antenne müsste diese Option vorsichtig und eingehend geprüft werden.

Alchemy – Eine explorative Playful Interaction

Setup

Der Prototyp wurde, wie auch ein mögliches Produkt, sehr modular und auch transportfähig konstruiert. Daher ist auch nicht viel nötig, um den Prototypen aufzusetzen. Voraussetzung ist lediglich ein handelsübliches Micro-USB 2.0 Kabel, mit einem USB-A Stecker an einen Micro-B Stecker. Solche Kabel sind am besten bekannt als Ladekabel für Smartphones.

Ist ein solches Kabel zur Hand, und hat der Prototyp einen sicheren Stand nahe an einem USB-Hub, so öffnen Sie den hinteren, schrägen Teil der Box. Es wird empfohlen, die (weiße) Wand an dem der Rundkolben befestigt ist fest zu halten und dann den oberen Teil der Abdeckung weg zu ziehen, damit sich der Klettverschluss, der zur Befestigung verwendet wurde, einfach und problemlos löst. Die Abdeckung wurde mit einem Gelenkscharnier an der Basis des Prototypen festgemacht.

Machen sie nun den Arduino Micro ausfindig. Dabei handelt es sich um einen Microcontroller, auf welchem das Programm des Prototypen gespeichert ist. Dieser muss mit Strom versorgt werden, also stecken sie das Micro-B-Ende ihres Kabels an den Arduino, und den USB-A-Stecker an einen mit Strom versorgten USB-Hub.

Nun können Sie die Abdeckung wieder schließen. Keine Bange, es wurde auf einen ausreichenden Abstand zwischen der Abdeckung und der Basis geachtet, sie können das Kabel also nicht einquetschen wenn Sie es zwischen Abdeckung und Basis entlangführen.

Damit ist Alchemy bereits einsatzbereit!

Bedienungsanleitung

Wenn Sie die Stromversorgung sicher gestellt haben, flackern die LED's weißlich, um zu signalisieren, dass der Prototyp einsatzbereit ist. Sie können nun eine beliebige der Küvetten nehmen und halten den blauen Deckel an über den Hals der Flasche, ganz so, als wollten sie eine darin befindliche Flüssigkeit in den Rundkolben schütten. Nun müssten die LED's derart leuchten, dass der Eindruck einer nach unten fließenden Flüssigkeit entsteht. Falls das nicht passiert, haben Sie etwas Geduld und versuchen sie mehrmals eine Flüssigkeit in den Rundkolben zu schütten; der verwendete RFID-Sensor erkennt manchmal den Tag nicht sofort. Sobald die Flüssigkeit im Rundkolben ist, können Sie eine *andere* Küvette verwenden, und dessen „Inhalt“ ebenso in den Rundkolben „schütten“. Die beiden Flüssigkeiten vermischen sich dann. Fügen Sie nun auf dieselbe Art und Weise eine dritte Flüssigkeit hinzu, erhalten Sie entweder nur eine „Flüssigkeit“ in der Farbmischung der drei verwendeten Ingredienzien, oder, falls eine von vielen bestimmten Kombinationen verwendet wurde, ein Bild. Beispielsweise können so Regenbögen oder Gewitter in der Flasche „erschaffen“ werden.

Diese Mischflüssigkeit oder das Bild wird für einige Sekunden angezeigt, ehe der Inhalt in der Flasche durch rot aufblinkende „verbrannt“ wird. Dann kann der Prozess von neuem gestartet werden

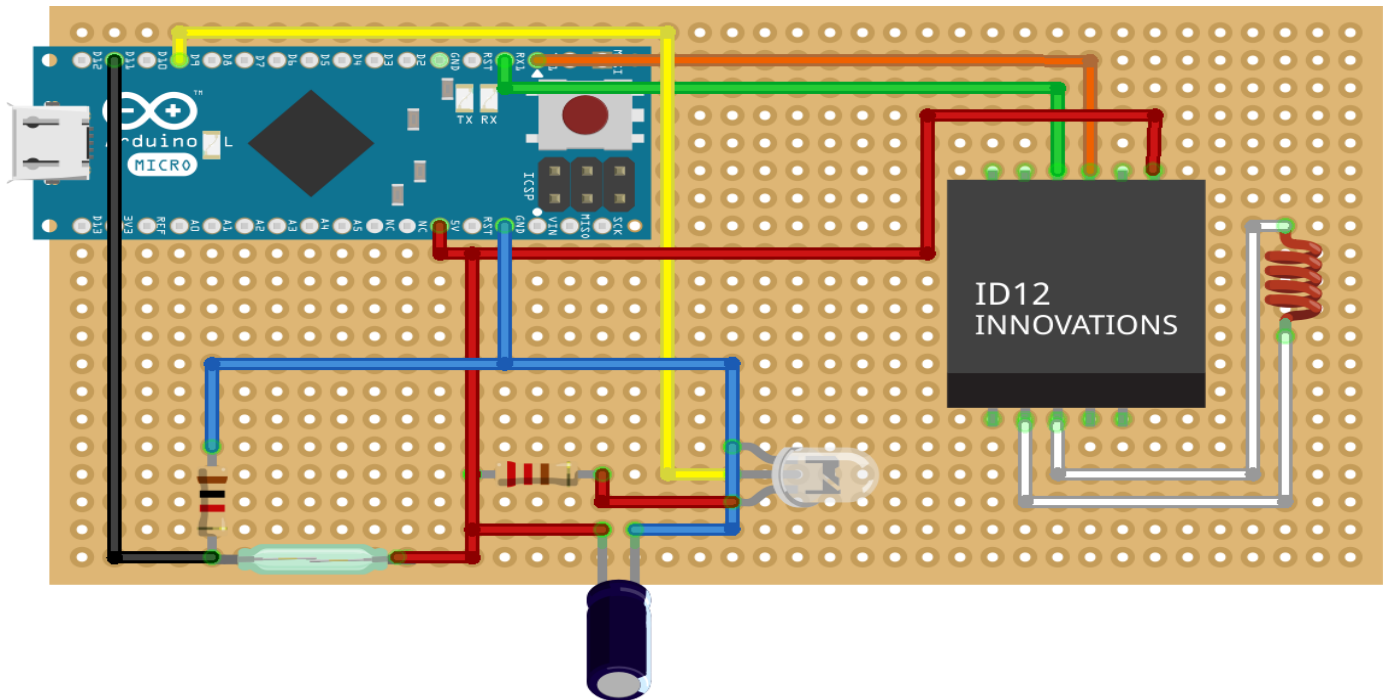
Falls Sie zu irgendeinem beliebigen Zeitpunkt denken, sich vertan zu haben und eine falsche Flüssigkeit verwendet zu haben, können Sie an dem Seil rechts des Rundkolbens ziehen. Dieses System nennen wir liebevoll „die Spülung“. Hierdurch wird die Flasche erneut durch das „Feuer“ gereinigt und Sie können von vorne beginnen.

Alchemy – Eine explorative Playful Interaction

Konzept

Ausführlichere Beschreibung des Interaktionskonzeptes

Implementierung



fritzing

Die Implementierung von „Alchemy“ ist seitens der Hardware wenig komplex.

Herzstück des Prototypen stellt der Arduino Micro dar, welcher mit einem USB-Kabel mit Strom versorgt wird. Es wird jedoch dringend empfohlen während des Aufbaus der Schaltung den Arduino noch nicht am Strom angeschlossen zu haben! Stellen Sie erst sicher, dass die Schaltung korrekt aufgebaut wurde.

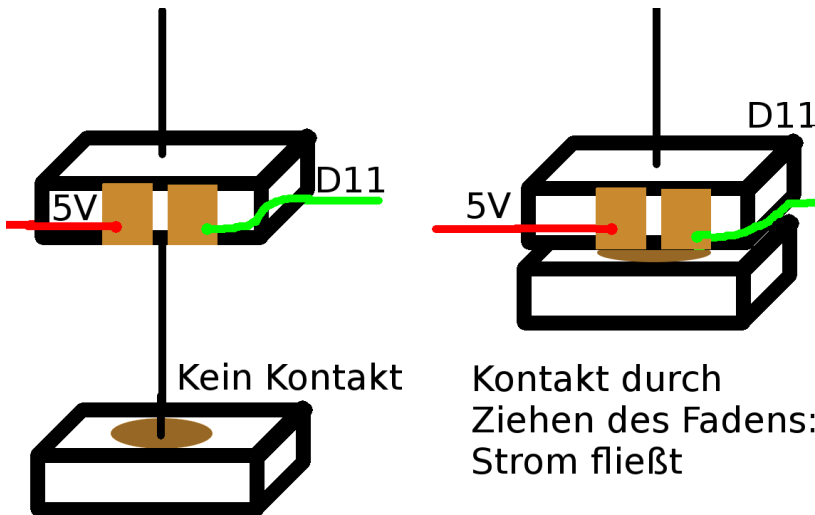
Als primärer Sensor wurde ein RFID-Reader des Modells [125Khz RFID module RDM6300 - UART](#) verwendet. Es kann jedoch jeder beliebige, andere RFID-Reader verwendet werden. Hierbei müssen Sie allerdings darauf achten, die richtigen Pins, die zu ihrem Modell passen, zu verwenden.

In unserem Fall wurden, zusätzlich zur grundlegenden Stromversorgung, die Pins RX und TX mit den entsprechenden Datenpins des Sensors verbunden. Schließlich benötigt der Reader noch eine „Antenne“, die ebenfalls an den Reader angeschlossen ist.

Als weiteren Sensor verwendeten wir eine selbst erstellte Konstruktion, welche im Grunde nur als Button fungiert. Wie bei jedem üblichen Button wird das eine Ende des Buttons mit 5V versorgt, das andere Ende wird an einen Datenpin gelegt, welcher auf das Signal hört. Wir verwendeten im Prototypen dafür den Pin D11. Da allerdings, sobald der Button einmal gedrückt ist, Strom am Pin anliegt und daher stets als „aktiviert“ beziehungsweise „1“ (entspricht „Strom liegt an“) erkannt wird, benötigt man eine Verbindung des Buttons mit GND. Damit der Strom aber nicht nur über den geringen Widerstand des GND fließt, bringt

man hier zwischen GND und Button noch einen hohen Widerstand ein (ideal sind $1k\Omega$). Wird der Button nun gedrückt, erkennt der Arduino dass der Button gedrückt wurde; und etwaige „Restströme“ fließen dann über den Ground wieder ab.

Bisher wurde vereinfacht von einem „Button“ gesprochen. Die eigentliche Konstruktion besteht aus zwei auf ein Holzstück geklebte Kupferstreifen, die keinen Kontakt haben, an denen jedoch jeweils Kabel gelötet wurden, welche zu den 5V- und D11-Pins führen. Darunter befindet sich ein weiteres Holzstück, auf dem ein leitendes Objekt angebracht ist, das den Kontakt zwischen den beiden Kupferstreifen schließen kann, wenn man an einem Faden zieht, welches am unteren Holzstück festgemacht ist und durch ein Loch im oberen Holzstück führt. Zu guter Letzt wurden noch [LED-Strips von Adafruit](#) verwendet. Da diese sehr empfindlich



sind, wurde weiterhin ein $47\mu F$ Kondensator parallel geschaltet, um die LEDs vor plötzlichen Stromschwankungen zu schützen. Auch wird ein Vorwiderstand von 220Ω empfohlen, um sicher zu gehen, dass die angelegte Spannung nicht zu hoch ist und die LEDs sich so erhitzen oder sogar direkt durchbrennen. Schließlich, sobald diese Sicherheitsvorkehrungen erfüllt sind, werden die LEDs an 5V- und GND-Pins angeschlossen, sowie an einen D9-Datenpin, welcher die LEDs ansteuert.

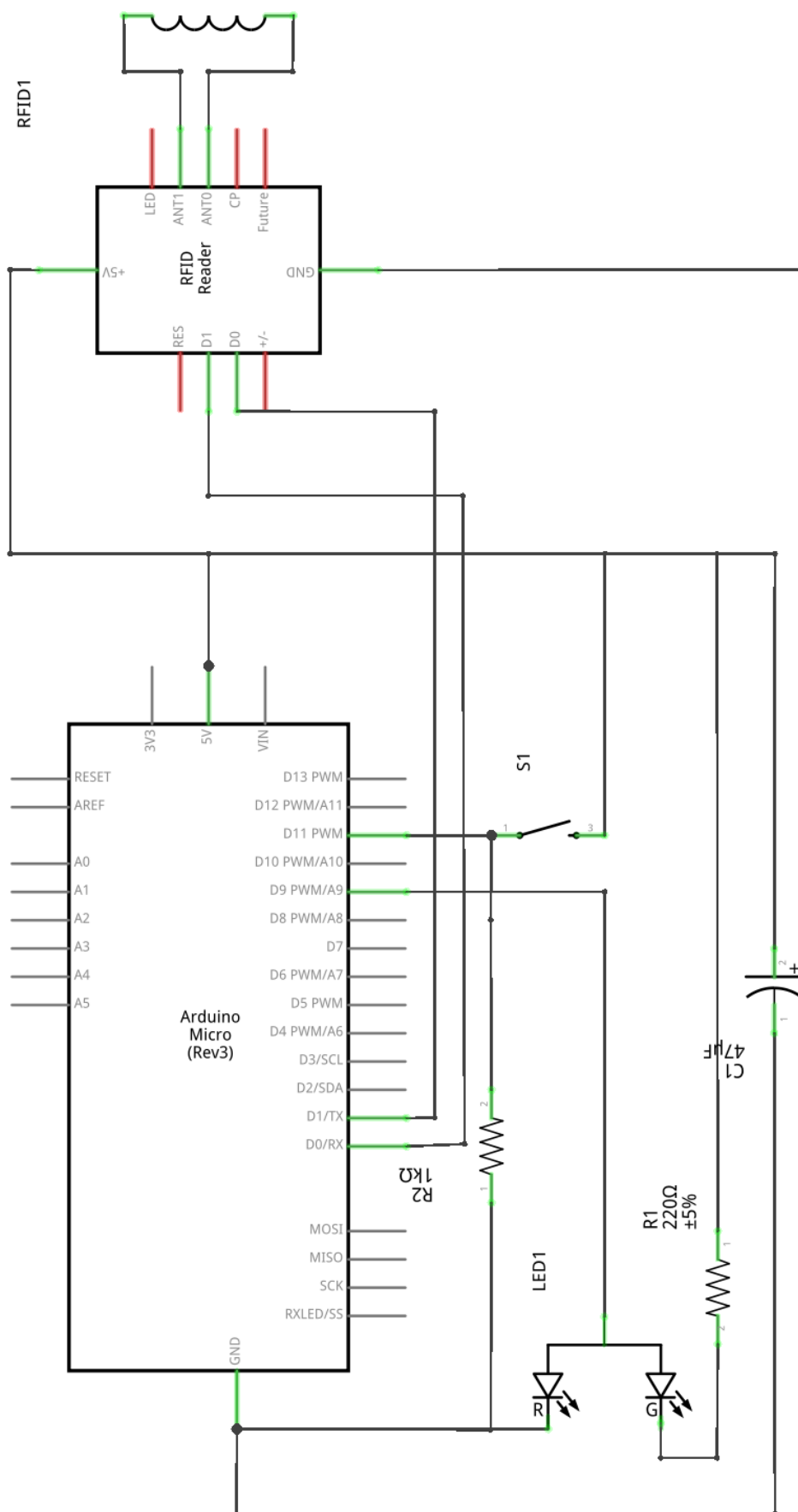
Im Fall unseren Prototypen verwenden wir 44 LEDs, welche in Reihe geschaltet sind. Diese werden dann im Programm mit Hilfe einer Matrix gesteuert.

Zu guter Letzt werden die RFID-Tags an die Deckel der Küvetten angebracht. Wir raten dabei besondere Vorsicht walten zu lassen! Die Tags können sehr empfindlich gegenüber Druck reagieren und ihre Funktion verlieren.

L1

fritzing

Part1



Alchemy – Eine explorative Playful Interaction

Code: main.ino

```
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h>
#endif
#define PIN_LED 9
#define PIN_RESET 11
#define N_LEDS 44
#define __ 255
#define SIZE 8
#define QUEUE 10
#define VANISH_TIME 200
/*
 * The pictures converted by the python script
 */
uint8_t picture_flower[] =
{0,0,0,0,0,0,0,0,0,0,0,0,82,255,0,0,0,0,82,255,0,0,0,0,0,0,0,0,255,0,50,0,0,
,0,0,0,0,0,0,0,0,0,0,0,0,82,255,0,255,0,0,82,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,0,0,0,0,0,0,255,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,255,0,0,0,0};
uint8_t picture_rainbow[] =
{0,0,0,0,0,0,255,0,50,255,0,50,255,0,50,255,91,0,255,91,0,255,0,50,255,0,50,255
,91,0,255,255,0,255,255,0,255,91,0,255,0,50,255,0,50,255,91,0,255,255,0,0,255,0
,0,255,0,255,255,0,255,91,0,255,0,50,255,91,0,255,255,0,0,255,0,0,82,255,0,82,2
55,0,255,0,255,255,0,255,91,0,255,255,0,0,255,0,0,82,255,255,0,255,255,0,255,0,
82,255,0,255,0,255,255,0,0,82,255,255,0,255,0,0,0,0,0,255,0,255,0,82,255};
uint8_t picture_thunder[] =
{63,63,116,63,63,116,91,110,225,91,110,225,99,155,255,99,155,255,99,155,255,99,
155,255,0,0,0,0,0,0,0,0,0,0,251,242,54,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,251,24
2,54,251,242,54,0,0,0,0,0,0,0,0,0,0,0,0,251,242,54,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,251,242,54,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,251,24
2,54,0,0,0,0,0,0};
uint8_t picture_fireflies[] =
{0,0,0,0,0,0,192,229,0,0,0,0,0,0,0,0,0,0,0,0,192,229,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,192,229,0,0,0,0,192,229,0,0,0,0,192,229,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,192,229,0,0,0,0,192,229,0,0,0,0,0,0,0,0,0,0,0,
0,192,229,0,0,0,0,0,0,0,0,0,0,0,0,192,229,0,0,0,0,0,0,0,0,0,0,0,0};
uint8_t picture_water[] =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,85,85,170,170,170,255,170,170,
255,85,85,170,85,85,170,170,170,255,170,170,255,170,170,255,85,85,170,85,85,170
,170,170,255,170,170,255,85,85,170,85,85,170,85,85,170,85,85,170,170,170,255,17
0,170,255,85,85,170,85,85,170,170,170,255,170,170,255,170,170,255,170,170,255,8
5,85,170,85,85,170,170,170,255,170,170,255,85,85,170,85,85,170,85,85,170,170,17
0,255,170,170,255,85,85,170,85,85,170,170,170,255};

int counter = 0;

uint16_t glowIntensity = 0;

uint16_t mixedElementsNum = 0;
```

```
boolean rendering = true;

int totalyMixed = 0;

boolean glowing = true;

int vanishCounter = 0;

int glowIncreaseDirection = 1;

uint32_t colors[5];

/*
 * Superclass from which other classes will inherit
 */
class Element {
public:
    virtual int get_r() = 0;
    virtual int get_g() = 0;
    virtual int get_b() = 0;
    virtual String get_id() = 0;
    virtual char get_char() = 0;
    virtual int get_num() = 0;
    virtual boolean get_used() = 0;
    virtual void set_used() = 0;
};

class Fire : public Element{
public:
    int get_r() { return color_r; };
    int get_g() { return color_g; };
    int get_b() { return color_b; };
    String get_id() { return id; };
    char get_char() { return c; };
    int get_num() { return num; };
    boolean get_used() { return used; };
    void set_used() { used = true; };
private:
    int color_r = 255;
    int color_g = 0;
    int color_b = 0;
    String id = "#0F005BD2E660#"; //0006017766
    char c = 'f';
    int num = 1;
    boolean used = false;
};Code: main.ino
#include <Adafruit_NeoPixel.h>
#ifdef __AVR__
#include <avr/power.h>
#endif

#define PIN_LED 9
#define PIN_RESET 11
#define N_LEDS 44
#define __ 255
#define SIZE 8
#define QUEUE 10
#define VANISH_TIME 200
```



```
/*
 * The pictures converted by the python script
 */
uint8_t picture_flower[] =
{0,0,0,0,0,0,0,0,0,0,0,0,82,255,0,0,0,0,82,255,0,0,0,0,0,0,0,0,0,0,255,0,50,0,0,
,0,0,0,0,0,0,0,0,0,0,0,0,0,82,255,0,255,0,0,82,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,255,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,255,0,0,0,0};
uint8_t picture_rainbow[] =
{0,0,0,0,0,0,255,0,50,255,0,50,255,0,50,255,91,0,255,91,0,255,0,50,255,0,50,255
,91,0,255,255,0,255,255,0,255,91,0,255,0,50,255,0,50,255,91,0,255,255,0,0,255,0
,0,255,0,255,255,0,255,91,0,255,0,50,255,91,0,255,255,0,0,255,0,0,82,255,0,82,2
55,0,255,0,255,255,0,255,91,0,255,255,0,0,255,0,0,82,255,255,0,255,255,0,255,0,
82,255,0,255,0,255,255,0,0,82,255,255,0,255,0,0,0,0,0,0,255,0,255,0,82,255};
uint8_t picture_thunder[] =
{63,63,116,63,63,116,91,110,225,91,110,225,99,155,255,99,155,255,99,155,255,99,
155,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,251,242,54,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,251,24
2,54,251,242,54,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,251,242,54,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,251,242,54,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,251,24
2,54,0,0,0,0,0,0};
uint8_t picture_fireflies[] =
{0,0,0,0,0,0,192,229,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,192,229,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,192,229,0,0,0,0,192,229,0,0,0,0,192,229,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,192,229,0,0,0,0,192,229,0,0,0,0,0,0,0,0,0,0,0,
0,192,229,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,192,229,0,0,0,0,0,0,0,0,0,0};
uint8_t picture_water[] =
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,85,85,170,170,170,255,170,170,
255,85,85,170,85,85,170,170,170,255,170,170,255,170,170,255,85,85,170,85,85,170
,170,170,255,170,170,255,85,85,170,85,85,170,85,85,170,85,85,170,170,170,255,17
0,170,255,85,85,170,85,85,170,170,170,255,170,170,255,170,170,255,170,170,255,8
5,85,170,85,85,170,170,170,255,170,170,255,85,85,170,85,85,170,85,85,170,170,17
0,255,170,170,255,85,85,170,85,85,170,170,170,255};

int counter = 0;

uint16_t glowIntensity = 0;

uint16_t mixedElementsNum = 0;

boolean rendering = true;

int totalyMixed = 0;

boolean glowing = true;

int vanishCounter = 0;

int glowIncreaseDirection = 1;

uint32_t colors[5];

/*
 * Superclass from which other classes will inherit
 */
class Element {
public:
    virtual int get_r() = 0;
    virtual int get_g() = 0;
```



```
virtual int get_b() = 0;
virtual String get_id() = 0;
virtual char get_char() = 0;
virtual int get_num() = 0;
virtual boolean get_used() = 0;
virtual void set_used() = 0;
};

class Fire : public Element{
public:
    int get_r() { return color_r;};
    int get_g() { return color_g;};
    int get_b() { return color_b;};
    String get_id() { return id;};
    char get_char() {return c;};
    int get_num() {return num;};
    boolean get_used() {return used;};
    void set_used() {used = true
};

class Water : public Element{
public:
    int get_r() { return color_r;};
    int get_g() { return color_g;};
    int get_b() { return color_b;};
    String get_id() { return id;};
    char get_char() {return c;};
    int get_num() {return num;};
    boolean get_used() {return used;};
    void set_used() {used = true;};
private:
    int color_r = 0;
    int color_g = 0;
    int color_b = 255;
    String id = "#0F005C174004#"; //0006035264
    char c = 'w';
    int num = 20;
    boolean used = false;
};

/*
 * "Soil" in code but later renamed to "Life"
 */
class Soil : public Element{
public:
    int get_r() { return color_r;};
    int get_g() { return color_g;};
    int get_b() { return color_b;};
    String get_id() { return id;};
    char get_char() {return c;};
    int get_num() {return num;};
    boolean get_used() {return used;};
    void set_used() {used = true;};
private:
    int color_r = 0;
    int color_g = 255;
    int color_b = 0;
    String id = "#0F005BB3A146#"; //0006009761
    char c = 's';
    int num = 300;
```

```
    boolean used = false;
};

class Air : public Element{
public:
    int get_r() { return color_r; };
    int get_g() { return color_g; };
    int get_b() { return color_b; };
    String get_id() { return id; };
    char get_char() { return c; };
    int get_num() { return num; };
    boolean get_used() { return used; };
    void set_used() { used = true; };
private:
    int color_r = 255;
    int color_g = 255;
    int color_b = 255;
    String id = "#0F005CE52791#"; //0006087975
    char c = 'a';
    int num = 4000;
    boolean used = false;
};

class Lightning : public Element{
public:
    int get_r() { return color_r; };
    int get_g() { return color_g; };
    int get_b() { return color_b; };
    String get_id() { return id; };
    char get_char() { return c; };
    int get_num() { return num; };
    boolean get_used() { return used; };
    void set_used() { used = true; };
private:
    int color_r = 255;
    int color_g = 255;
    int color_b = 0;
    String id = "#0F005CB98C66#"; //0006076812
    char c = 'l';
    int num = 50000;
    boolean used = false;
};

/*
 * Every element must be registered here to work
 */
Element* allElements[] = {new Fire(), new Water(), new Soil(), new Air(), new
Lightning()};

boolean activeFilling = false;

/*
 * Matrix where every led has ist own id
 */
static int leds [SIZE][SIZE] = {
    { __, __, __, 14, 29, __, __, __ },
    { __, __, __, 15, 28, __, __, __ },
    { __, __, 13, 16, 27, 30, __, __ },
```

```
{ __, 3, 12, 17, 26, 31, 40, __ },
{ 2, 4, 11, 18, 25, 32, 39, 41 },
{ 1, 5, 10, 19, 24, 33, 38, 42 },
{ 0, 6, 9, 20, 23, 34, 37, 43 },
{ __, 7, 8, 21, 22, 35, 36, __ }
};

/*
 * Drop struct which is used for fluid simulation
 */
struct Drop {
public:
    int x;
    int y;
    int color_r = 0;
    int color_g = 0;
    int color_b = 0;
    int newColor_r = -1;
    int newColor_g = -1;
    int newColor_b = -1;
    boolean rendered = false;
    int sinceMoved = 0;
    int action = 0;
};

Adafruit_NeoPixel strip = Adafruit_NeoPixel(N_LEDS, PIN_LED, NEO_GRB +
NEO_KHZ800);

/*
 * Drop array where drops are stored
 */
Drop* drops[SIZE][SIZE];

Element* actualElement;

int actualElementNum = 0;

/*
 * Standby glow effect
 */
void glow() {

    glowIntensity += glowIncreaseDirection;

    if(glowIntensity >= 1023 || glowIntensity <= 0) {
        glowIncreaseDirection *= -1;
    }

    for(int i = 0; i < N_LEDS; i++) {
        uint32_t c = strip.Color(glowIntensity/8, glowIntensity/8, glowIntensity/8);
        strip.setPixelColor(i, c);
    }

    strip.show();
}
```

```
/*
 * Creates a fire effect
 */
void vanish() {
    while(true) {
        if(vanishCounter > VANISH_TIME) {
            softwareReset();
        }

        for(int i = 0; i < N_LEDS; i++) {
            uint32_t c = 0;
            int r = random(0, 8);
            if(r < 1) {
                c = strip.Color(255, 0, 0);
            }
            strip.setPixelColor(i, c);
        }

        vanishCounter++;

        strip.show();
    }
}

/*
 * Is called on every drop every update
 */
void render(Drop* drop) {
    if (drop->rendered) {
        return;
    }
    int x = drop->x;
    int y = drop->y;
    if (x >= SIZE-1) {
        return;
    }
    if (leds[x + 1][y] != 255) {
        if (drops[x + 1][y] == NULL) {
            drops[x][y] = NULL;
            drop->x = x + 1;
            drops[drop->x][drop->y] = drop;
        } else {
            push(drops[x + 1][y]);
        }
    }
    renderColors(drop);
}

/*
 * Is called if a drop tries to push another away to fall down
 */
void push(Drop* drop) {
    if (drop->rendered) {
        return;
    }
    if (drop->sinceMoved > 0) {
        drop->sinceMoved = 0;
        return;
    }
}
```

```
drop->sinceMoved += 1;
int x = drop->x;
int y = drop->y;

int r = random(0, 2);

if (r == 0) {
    pushRight(drop);
} else {
    pushLeft(drop);
}
}

/*
 * Tries to push the drop to the left
 */
void pushLeft(Drop* drop) {
    int x = drop->x;
    int y = drop->y;
    if (drop->action > 0) {
        return;
    }
    if (y - 1 < 0) {
        return;
    }
    if (leds[x][y - 1] != 255) {
        if (drops[x][y - 1] == NULL) {
            drops[x][y] = NULL;
            drop->y = y - 1;
            drops[drop->x][drop->y] = drop;
            return;
        }
        else {
            push(drops[x][y - 1]);
        }
    }
    drop->action = drop->action + 1;
    pushRight(drop);
}

/*
 * Tries to push the drop to the right
 */
void pushRight(Drop* drop) {
    int x = drop->x;
    int y = drop->y;
    if (drop->action > 0) {
        return;
    }
    if (y + 1 >= SIZE) {
        return;
    }
    if (leds[x][y + 1] != 255) {
        if (drops[x][y + 1] == NULL) {
            drops[x][y] = NULL;
            drop->y = y + 1;
            drops[drop->x][drop->y] = drop;
        }
    }
}
```

```
        return;
    } else {
        push(drops[x][y + 1]);
    }
}
drop->action = drop->action + 1;
pushLeft(drop);
}

/*
 * Is called for every drop every update
 */
void renderColors(Drop* drop) {
    float r = drop->color_r * 50;
    float g = drop->color_g * 50;
    float b = drop->color_b * 50;
    int x = drop->x;
    int y = drop->y;
    float counter = 50.0;

    for (int i = -1; i <= 1; i++) {
        for (int j = -1; j <= 1; j++) {
            int newX = x + i;
            int newY = y + j;
            if (newX >= 0 && newX < SIZE) {
                if (newY >= 0 && newY < SIZE) {
                    if (drops[newX][newY] != NULL) {
                        Drop* newDrop = drops[newX][newY];
                        r += (float)newDrop->color_r;
                        g += (float)newDrop->color_g;
                        b += (float)newDrop->color_b;
                        counter += 1.0;
                    }
                }
            }
        }
    }
    r = r / counter;
    g = g / counter;
    b = b / counter;

    drop->newColor_r = (int)round(r);
    drop->newColor_g = (int)round(g);
    drop->newColor_b = (int)round(b);
}

void setup() {

    Serial.begin(9600);
    Serial1.begin(9600);
    pinMode(PIN_RESET, INPUT);

    colors[0] = strip.Color(0, 80, 0);
    colors[1] = strip.Color(80, 0, 0);
    colors[2] = strip.Color(0, 0, 80);
    colors[3] = strip.Color(80, 0, 80);
    colors[4] = strip.Color(0, 80, 80);

    strip.begin();
```

```
for (int x = 0; x < SIZE; x++) {
  for (int y = 0; y < SIZE; y++) {
    drops[x][y] = NULL;
  }
}

void updateColors(Drop* drop) {
  if (drop->newColor_r != -1) {
    drop->color_r = drop->newColor_r;
    drop->newColor_r = -1;
  }
  if (drop->newColor_g != -1) {
    drop->color_g = drop->newColor_g;
    drop->newColor_g = -1;
  }
  if (drop->newColor_b != -1) {
    drop->color_b = drop->newColor_b;
    drop->newColor_b = -1;
  }
}

/*
 * Creates 10 new drops with the given element
 */
void fill(Element* element) {
  if(element->get_used()){
    return;
  }
  Serial.println("Actual Elemets: ");
  mixedElementsNum += element->get_num();
  element->set_used();
  actualElementNum = 10;
  actualElement = element;
  totalyMixed++;
}

/*
 * Shows given picture for 5 seconds and then calls vanish()
 */
void showPicture(uint8_t picture[]){
  resetLEDs();
  Serial.println("showing picture");
  rendering = false;

  int counter = 0;

  for(int x = 0; x < SIZE; x++){
    for(int z = 0; z < SIZE; z++){
      if(leds[x][z] == __){
        continue;
      }
      int rgb[3];
      for(int i = 0; i < 3; i++){
        rgb[i] = picture[counter];
        counter++;
      }
      int ledNum = leds[x][z];
```



```
int color = picture[counter];

int r = rgb[0];
int g = rgb[1];
int b = rgb[2];
uint32_t c = strip.Color(r, g, b);
strip.setPixelColor(ledNum, c);
}
}

strip.show();

delay(5000);

vanish();
}

/*
 * Turn every LED off
 */
void resetLEDs() {
  for(int i = 0; i < N_LEDS; i++) {
    uint32_t c = strip.Color(0, 0, 0);
    strip.setPixelColor(i, c);
    strip.show();
  }
}

/*
 * Read input from RFID chip
 */
void readSerial() {
  if (Serial1.available() > 0) {
    String incomingByte = Serial1.readString();

    Element* element = findElement(incomingByte);
    if(element != NULL) {
      glowing = false;
      resetLEDs();
      Serial.println("found");
      fill(element);
    }
  }
}

/*
 * Returns the element with the given ID
 */
Element* findElement(String id) {
  for(int i = 0; i < 5; i++) {
    Element* element = allElements[i];
    String actualId = element->get_id();
    if(actualId.equals(id)) {
      activeFilling = true;
      return element;
    }
  }
}
```

```
    return NULL;
}

/*
 * If 3 elements are combined, this function will check if it is a valid
 * combination and show a picture
 */
void combine() {
    if(totallyMixed < 3) {
        return;
    }

    if(mixedElementsNum == 321) {
        showPicture(picture_flower);
    } else if(mixedElementsNum == 54020) {
        showPicture(picture_rainbow);
    } else if(mixedElementsNum == 54001) {
        showPicture(picture_thunder);
    } else if(mixedElementsNum == 4320) {
        showPicture(picture_fireflies);
    } else {
        delay(5000);
        vanish();
    }
}

void loop() {
    // Checks if reset is pulled
    int input = digitalRead(PIN_RESET);
    if(input != 0) {
        vanish();
    }

    if(!rendering) {
        return;
    }

    if(!activeFilling) {
        readSerial();
    }

    if(glowing) {
        glow();
        return;
    }

    // creates new elements
    if(actualElementNum > 0) {
        int x = 0;
        int y = 3 + actualElementNum%2;
        Drop* drop = new Drop();
        drop->x = x;
        drop->y = y;
        drop->color_r = actualElement->get_r();
        drop->color_g = actualElement->get_g();
        drop->color_b = actualElement->get_b();
        drops[x][y] = drop;
        actualElementNum--;
    }
}
```

```
}else{
  activeFilling = false;
  combine();
}

//renders every drop
for (byte x = 0; x < SIZE; x++) {
  for (byte y = 0; y < SIZE; y++) {
    if (leds[x][y] != __) {
      if (drops[x][y] != NULL) {
        Drop* drop = drops[x][y];
        render(drop);
      }
    }
  }
}

//updates the color for every drop
for (byte x = 0; x < SIZE; x++) {
  for (byte y = 0; y < SIZE; y++) {
    if (leds[x][y] != __) {
      if (drops[x][y] != NULL) {
        Drop* drop = drops[x][y];
        updateColors(drop);
        byte led = leds[x][y];
        uint32_t c = strip.Color(drop->color_r, drop->color_g, drop->color_b);
        strip.setPixelColor(led, c);

        drop->rendered = false;
        drop->action = 0;
      }
    }
  }
}

//sets the color for the LEDs
for (byte x = 0; x < SIZE; x++) {
  for (byte y = 0; y < SIZE; y++) {
    int c = strip.Color(0, 0, 0);
    byte led = leds[x][y];
    if (led != __) {
      if (drops[x][y] == NULL) {
        strip.setPixelColor(led, c);
      }
    }
  }
}
strip.show();
}

void softwareReset() { //http://forum.arduino.cc/index.php?topic=49581.0
  asm volatile (" jmp 0");
}

Weiterer Code wie das Python-Skript zum Erstellen von Bildern kann auf
https://github.com/Lyniat/Alchemystry gefunden werden.
```