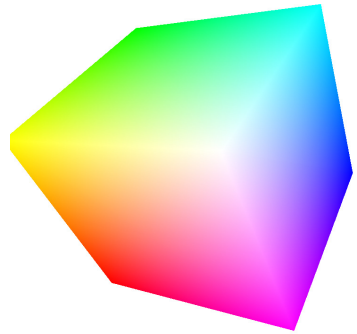
# LIGHTBOY
## EINE LED-LAMPE




### AUTOR

Mark Welch
Mittlerer Graben 20
86152 Augsburg



### ABSTRACT

Diese Studienarbeit befasst sich mit der Ausarbeitung einer Lampe, die auf RGB-LEDs und einem Arduino beruht.
Sie wird per Webtablet (hier iPad) gesteuert werden und soll möglichst viele verschiedene RGB-Werte anzeigen und auch sonst einige Extras bieten.



### CATEGORIES AND SUBJECT DESCRIPTORS

H.5.2 [User Interfaces]  Graphical user interfaces (GUI)
D.2.2 User Interfaces

### GENERAL TERMS

Human Factors, Design, Experimentation, Reliability, Performance

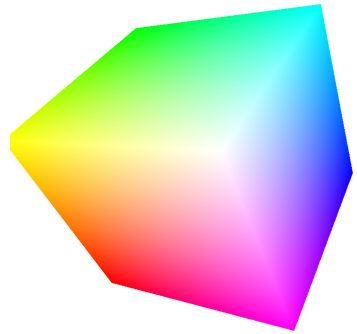
### KEYWORDS

HTML, Javascript, Serial Communication, USB, Steuerung, RGB, Serproxy


### I. MOTIVATION

Oft hat man das Problem, daß man Abends eine Lichtquelle braucht, aber normale Lampen zu hell sind oder nicht zum Ambiente passt. Daher wär es schön eine Lampe zu haben die man farblich seiner Stimmung oder der Umgebung anpassen kann.
Zudem wäre es schön, dass man die Lampe so konzipieren, dass man Farbmuster oder das Licht animiert leuchten lassen kann. So daß sie per Software auf  Musik reagiert und daran die Anzeige regelt.
Aufgrund der Farbkomplexität solcher Lampen ist es oft schwer sie nur über normale (Dreh-)Regler  zu steuern und dies schränkt die Möglichkeiten der Lampe auf ein kleines Minimum zusammen, daher wäre eine  Steuerung über ein ausführlicheres Grafikinterface positiv.
Professionelle Mood-Lights sind oft sehr teuer und haben eine nicht immer vorteilhafte Steuerung, daher wollte ich eine halbwegs günstige Lampe bauen, bzw eine die ich meinen Bedürfnissen anpassen  oder erweitern kann.

# LIGHTBOY
## EINE LED-LAMPE



## 2. VERWANDTE ARBEITEN

Eine der ersten Dinge, die man in Tutorials für das Arduino lernt, ist es wie man eine LED zum Leuchten bringt. Zudem hat man bei LEDs eine sehr schnelle Ressounance, ob eine Awendung funktioniert oder nicht, zudem sind LED oft einfach schön anzusehen.

Aus diesen Gründen gibt es im Internet schon viele individuelle Projekte, in welchen RGB Lampen gebaut wurden. Zudem gibt es bei Instructables.com jährlich einen Wettbewerb, welcher den Umgang und das kreative Einsetzen  von LEDs fördert. So zum Beispiel dieses Projekt, welches mich ursprünglich inspiriert hatte:

**2.1 LAMPDUINO** - Eine Lampe als Raumausteller in der eine 8x8 RGB LED Matrix realisiert ist. Diese wird mittels eines Computers gesteuert. Hierbei kann man eigene Muster anzeigen lassen oder dank einer Arduino-Bibliothek die Lichter zu Musik als Audio Spectrum VU Meter animieren.

http://www.instructables.com/id/Lampduino-an-8x8-RGB-Floor-Lamp/

**2.2 HUE-CONTROLLABLE RGB LED LAMP** - ein kleiner RGB-LED Versuch der zeigt wie man die Werte aus einem HUE-Farbfeld an die Lampe überträgt und diese dann dort anzeigen lassen kann. in der eine 8x8 RGB LED Matrix realisiert ist und mittels des Computers gesteuert wird.

http://www.arduino.cc/cgi-bin/yabb2/YaBB.pl?num=1207331496

Ansonsten sind es einfach zuviele weitere Projekte um sie hier aufzuführen, einen guten Überblick findet man auf folgende Seiten:

http://www.instructables.com  und http://arduino.cc/forum/index.php/board,6.0.html

## 3. ANWENDUNGSSZENARIO & USE CASES

### GRUNDSZENARIO (VORRAUSSETZUNG FÜR DIE ANDEREN)

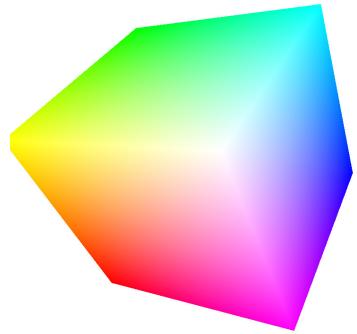
Akteur 1: Nutzer
Akteur 2: iPad
Akteur 3: Lampe

Vorraussetzung: Die Lampe ist an einer Stromquelle angeschlossen und der Webclient auf der Lampe läuft.

Anwendungsablauf: Der Nutzer nimmt das iPad und verbindet sich per Wi-Fi mit der Lampe hier navigiert er auf dem Interface um folgende Anwendungsfälle durzuführen.

### 3.1 SETZE FARBE DER LAMPE

Hier wählt er einen der folgenden Darstellungsmodi, sucht aus welcher LED-Block aktuell gesetzt werden soll und wählt die Farbe aus. Dies wiederholt er bis er alle gewünschten Farben eingestellt hat. Anschliessend drückt er den Button “Farbe übertragen” und überträgt so die gewählten Einstellungen vom iPad an die Lampe und diese schaltet die LED dann in diese Darstellung.



**Modus a)** setzen einer Farbe für die gesammte Matrix
**Modus b)** für jede Reihe in der Matrix wird eine andere Farbe gesetzt
**Modus c)** jede einzelne LED in der Matrix bekommt eine eigene Farbe

### 3.2 SCHALTE LAMPE AUS

Hier drückt der Nutzer, nach dem Standardmässigen Verbinden mit der Lampe, den Button "Anschalten". Die Lampe toggelt daraufhin zwischen An und Aus. Hierbei merkt er sich die Lampe, in welchem Darstellung wir uns zuvor befunden haben.

### 3.3 COLOR FADE

Hier wählt er den Modus "eigener Color-Fade" und wählt die zwei Farben aus, zwischen welche die Lampe hin und her interpoliert. Nachdem die Farben übertragen sind, wechselt die Lampe stufenweise in Zwischenfarben von der einen Farbe zur anderen und zurück. Dies geht solange bis der Darstellungsmodus gewechselt wird oder die Lampe ausgeschaltet wird.

### 3.4 TIC TAC TOE

Hier wählt der Nutzer den Modus "TicTacToe".
In diesem Modus wird der Nutzer aufgefordert eine Farbe als seine Spielerfarbe zu wählen. Nachdem dieser das getan hat wählt die Lampe ihre eigene Spielerfarbe.

Anschließend setzen Nutzer und Lampe jeweils abwechselnd eine LED auf ihre Farbe. Wie beim klassischen TicTacToe werden die verschiedenen Farben (so wie X und O) unterschieden.

Hat einer der Spieler drei der LEDs in seiner Reihe, Spalte oder Diagonale gewinnt er und die Lampe setzt die ganze Matrix in die Farbe des Gewinners. Bei einem Unentschieden bleibt die Lampe im Muster wie die LEDs zuletzt gesetzt wurden.

Anschliessend kann der Nutzer per Button "Spielfeld zurücksetzen" wählen ob er noch ein Spiel spielen will.

### 3.5 DEFAULTWERTE AN/AUS

Für jeden Darstellungsmodus wurden auf der Lampe Farbwerte als Default bereitgestellt.

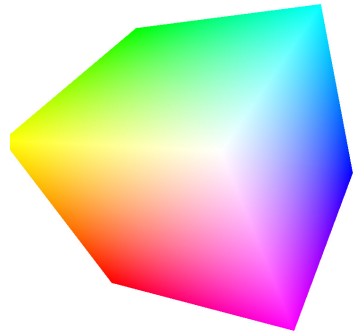
Über den Button "Defaultwerte an/aus" kann der Nutzer wählen, ob er die Defaultwerte nutzen will, oder lieber seine eigene Farben wählt.

## 4. ENTWURF EINES PROTOTYPEN

### 4.1 DAS IPAD

Für den LightBoy nutzen wir das iPad um eine Lampe zu Steuern. Das iPad ist ein Tablet mit Touchscreen, welches gern für das lean-back Multimedia Gefühl und entsprechende Anwendungen verwendet wird.

# LIGHTBOY
## EINE LED-LAMPE



Auf dem iPad kann man Programme auf zwei Methoden realisieren, entweder als native Applikation in der Programmiersprache Objective-C oder als Webanwendung (HTML+CSS)

Zudem hat das iPad auch eine WiFi, Bluetooth-Anbindungen, mit der man die Kommunikation zu einem Netzwerk oder einen Server realisieren kann.

### 4.2 KONZEPTIONELLER AUFBAU DES PROTOTYPEN

Für den Prototypen mussten wir zwei Teile realiseren, ein Interface zum Steuern der Anwendung, die Lampe/Hardware und die Programmierung dieser.

Mit dem iPad verbindet man sich per WiFi mit einem Server auf dem ein Interface läuft, über welches man die Einstellungen für die Lampe wählt. Anschliessend werden die Einstellungen vom Server an das Arduino übertragen, welches diese interpretiert und die Licht-Matrix auf unser gewähltes Muster schaltet.

# 5. IMPLEMENTIERUNG

## 5.1 SOFTWARE

### 5.1.1 INTERFACE

Das Interface habe ich über eine einfache HTML-Seite realisiert. Hier werden die nötigen Elemente die man zum einstellen braucht als HTML programmiert. Eine Tabelle mit den Farbwerten bietet die Basis zum wählen der LED-Farben. Nachdem man die Farben gewählt hat werden die Werte über jQuery und eine PHP-Seite an das Arduino übermittelt.

### 5.1.2 SERVER

Die Kommunikation mit dem Arduino wird über ein Webserver realisiert, auf dem eine HTML-Seite läuft, die die Farbwerte für die LED an das Arduino weitersendet.

Da ich leider kein Ethernetshield oder WiFI-Shield habe brauche ich einen PC auf dem der Webserver läuft, gleichzeitig läuft auf diesem auch das Programm “serproxy”. Über dieses Programm ist es möglich die COM-Ports als HTML-Proxy zu setzen und so mit dem Server die Werte an das Arduino zu übertragen. Dazu muss die Lampe (das Arduino) per USB-Kabel mit dem PC verbunden sein.

Mit einem Ethernet-Shield oder WiFi-Shield müssten wir den Umweg über den PC+USB nicht gehen und könnten den Webserver direkt auf dem Arduino laufen lassen. So würde man auch die Serielle Communication über das Programm “serproxy” vereinfachen, da das Arduino direkt vom Interface seine Werte kriegt.

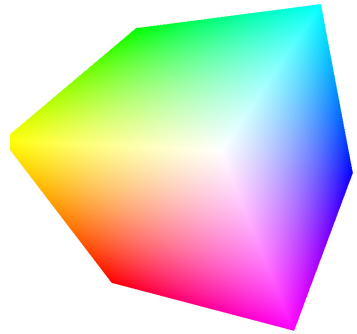
### 5.1.3 PROGRAMMIERUNG DER HARDWARE

Das Programm für die Hardware der Lampe ist in Processing realisiert, da dies die beste Programmierumgebung für das Arduino ist.

Im Programm ist ein Listener implementiert, der auf Signale am Serial Port wartet.

```
Serial.begin(115200);
```

# LIGHTBOY
## EINE LED-LAMPE



Diese interpretiert das Programm und wählt den richtigen Darstellungsmodus und setzt die richtige Farbe für die LEDs.
Um sicher zu gehen dass nur richtige Befehle interpretiert werden, und kein zufällig entstandener Datenmüll, wird ein Start und ein Endzeichen mit jedem gültigen Befehl mitgesendet.

Erst bei Empfang des Startzeichens wird das Sichern der Zeichen vom Serial Port gestartet und beim Endzeichen beendet.

```
void fetchCommand(){
   while(Serial.available() > 0){
       inChar = Serial.read();
      if(inChar == '(') {
          fcstart = true;
          fcend = false;
          clearCommand();
      }else if ( inChar == ')') {
          fcstart  = false;
          fcend = true;
          gotcmd = true;
      } else if ((fcstart) && !(fcend)){
          if(dat_ind < 69){
            inData[dat_ind] = inChar; // Store i
            dat_ind++;
            inData[dat_ind] = '\0';
          }
     }
  } //Serial.print(inData);
  //Serial.println(" infetch");
   // Null terminate the string*/
   inData[dat_ind] = '\0';
}
```

Anschließend wird der ZeichenString interpretiert, anhand der ersten beiden Zeichen wird der Zeichenmodus gewählt.

```
    fetchCommand();
    if(gotcmd){
        //checke ob die Nachricht mit den Zeichen "xx" beginnt
        if (inData[0] == 'z' && inData[1] == 'z'){
          ...
        } else if (inData[0] == 'y' && inData[1] == 'y'){
          ...
       }
       ...
   }
```

Nach dem wird der restliche String zerteilt und aus ihr die Farben, welche als Hex-Werte im String stehen aufgeteilt und in die einzelne Array für die LED-Matrix gesetzt werden:

```
...
        char color[7];
          int offset = 3;

          for (int c_int=0; c_int < 7; c_int++){
            color[c_int] = inData[(c_int+offset)];
          }
           color[7] = '\0';
           setColor( chosen_single_color, color );
           clearCommand();
...
```

Hierfür haben wir zwei Methoden, welche die hex-Werte in int-Werte übersetzt:

```
// umrechnung der einzelnen HexStellen in int-Werte
int convhex(char w) { ... }

int convert_hex(char farbe[]){
    byte w1 = convhex(farbe[0]);
    byte w2 = convhex(farbe[1]);

    if (w1<0 || w2<0) {  return 00;  }
    else { return int((w1*16) + w2); }
}
```

# LIGHTBOY
## EINE LED-LAMPE

### 5.1.4 DER TICTACTOE-CLIENT

Die Kernfunktion für den TicTacToe-Client ist im Web-Interface implementiert. Die Programmiersprache dafür ist JavaScript. Die Werte und der Spielstatus wird aus der Anwendung aus an die Lampe gesendet.

Methode die Checkt ob ein Spieler schon eine Gewinnreihe hat:

```
function find_winning_row() {
          var sp = 1;
          while ( (sp < 3) && (all==0) ){
                    if ((a==sp)&&(b==sp)&&(c==sp)) all=sp;
                    if ((a==sp)&&(d==sp)&&(g==sp)) all=sp;
                    if ((a==sp)&&(e==sp)&&(i==sp)) all=sp;
                    if ((b==sp)&&(e==sp)&&(h==sp)) all=sp;
                    if ((d==sp)&&(e==sp)&&(f==sp)) all=sp;
                    if ((g==sp)&&(h==sp)&&(i==sp)) all=sp;
                    if ((c==sp)&&(f==sp)&&(i==sp)) all=sp;
                    if ((g==sp)&&(e==sp)&&(c==sp)) all=sp;
                    sp++;
          }
          if ((a != 0)&&(b != 0)&&(c != 0)&&(d != 0)&&(e != 0)&&(f != 0)&&(g != 0)&&(h != 0)&&(i !=
0)&&(all == 0)) all = 3;
}
```

Methode um ein leeres Feld im Spiel zu finden:

```
function find_empty_spot() {

          if ( ((b == c)&&(a == 0)&&(temp==“”)) || ((d == g)&&(a == 0)&&(temp==“”)) || ((e ==
i)&&(a == 0)&&(temp==“”)) ) temp=”A”;
          if ( ((a == c)&&(b == 0)&&(temp==“”)) || ((h == e)&&(b == 0)&&(temp==“”)) ) temp=”B”;
          if ( ((a == b)&&(c == 0)&&(temp==“”)) || ((i == f)&&(c == 0)&&(temp==“”)) || ((g ==
e)&&(c == 0)&&(temp==“”)) ) temp=”C”;
          if ( ((a == g)&&(d == 0)&&(temp==“”)) || ((e == f)&&(d == 0)&&(temp==“”)) ) temp=”D”;
          if ( ((a == i)&&(e == 0)&&(temp==“”)) || ((b == h)&&(e == 0)&&(temp==“”)) || ((d ==
e)&&(f == 0)&&(temp==“”)) || ((g == c)&&(e == 0)&&(temp==“”)) ) temp=”E”;
          if ( ((d == e)&&(f == 0)&&(temp==“”)) || ((c == i)&&(f == 0)&&(temp==“”)) ) temp=”F”;
          if ( ((a == d)&&(g == 0)&&(temp==“”)) || ((h == i)&&(g == 0)&&(temp==“”)) || ((e ==
c)&&(g == 0)&&(temp==“”)) ) temp=”G”;
          if ( ((b == e)&&(h == 0)&&(temp==“”)) || ((g == i)&&(h == 0)&&(temp==“”)) ) temp=”H”;
          if ( ((a == e)&&(i == 0)&&(temp==“”)) || ((g == h)&&(i == 0)&&(temp==“”)) || ((c ==
f)&&(i == 0)&&(temp==“”)) ) temp=”I”;
}
```

Methode um zu Checken ob das Feld wirklich leer ist:

```
function check_chosen_spot() {
          if ((temp==”A”)&&(a==0)) {
                    ok=1;
                    a = (cf+1);
          } else if ((temp==”B”)&&(b==0)) {
                    ok=1;
                    b = (cf+1);
          } else if ((temp==”C”)&&(c==0)) {
                    ok=1;
                    c = (cf+1);
          } else if ((temp==”D”)&&(d==0)) {
                    ok=1;
                    d = (cf+1);
          } else if ((temp==”E”)&&(e==0)) {
                    ok=1;
                    e = (cf+1);
          } else if ((temp==”F”)&&(f==0)) {
                    ok=1
                    f = (cf+1);
          } else if ((temp==”G”)&&(g==0)) {
                    ok=1
                    g = (cf+1);
          } else if ((temp==”H”)&&(h==0)) {
                    ok=1;
                    h = (cf+1);
          } else if ((temp==”I”)&&(i==0)) {
                    ok=1;
                    i = (cf+1);
          }
}
```

Methode um den Computer zu seinem Zug zu verhelfen:

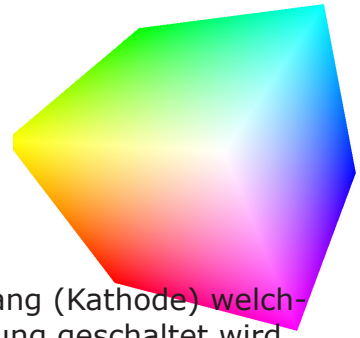
```
function computerChoice() {
          temp=””;
          ok = 0;
          cf = 1;
          find_empty_spot();
          check_chosen_spot();

          while(ok==0) {
                    aRandomNumber=Math.random()
                    comp=Math.round((choice-1)*aRandomNumber)+1;
                    if (comp==1) temp=”A”;
                    else if (comp==2) temp=”B”;
                    else if (comp==3) temp=”C”;
                    else if (comp==4) temp=”D”;
                    else if (comp==5) temp=”E”;
                    else if (comp==6) temp=”F”;
                    else if (comp==7) temp=”G”;
                    else if (comp==8) temp=”H”;
                    else if (comp==9) temp=”I”;
                    check_chosen_spot();
          }

          $(‘.ttt-led[rev=”’+temp+’”]’).attr(‘rel’, $(“#player2”).attr(‘rel’)).css(“background-
color”, $(“#player2”).attr(‘rel’));
          process();

          if (all==0) { setcolor(); }
}
```

# LIGHTBOY
## EINE LED-LAMPE



### 5.2 DIE HARDWARE

#### 5.2.1 SERVER

Da leider für die Entwicklung kein Ethernetshield oder WiFi-Shield zur Verfügung stand musste ich eine andere Methode wählen, um die Kommunikation zwischen Lampe und iPad zum Testen zu realisieren. Daher kann man die Lampe erstmal nicht als Standalone-Gerät im Raum installieren, aber so konnte man schonmal den Rest realisieren und anschliessend das Gerät bei vorhanden sein mit einem Ethernet oder WiFi-Shield erweitern.

#### 5.2.2 ARDUINO BASIS FÜR DIE LAMPE

Für die Hardware nutzen wir als Basis ein Arduino.
Dies ist ein einfacher Steuerkontroller, welcher gut für Prototypentwicklung genutzt werden kann, da man hier leicht ohne Programmierung eine Schaltung aufbauen kann, ganz ohne löten, so kann immer schnell was korrigiert werden. Zudem ist der Arduino leicht zu programmieren. Hierfür nimmt man eine Entwicklerumgebung, welche auf Processing beruht.

Für das Arduino gibt es auch einige Erweiterungsteile, so genannte Shields. Mit ihnen lässt sich den Funktionsumfang erweitern. Zudem lässt sich das Arduino leicht über die Programmiersprache Processing programmieren.

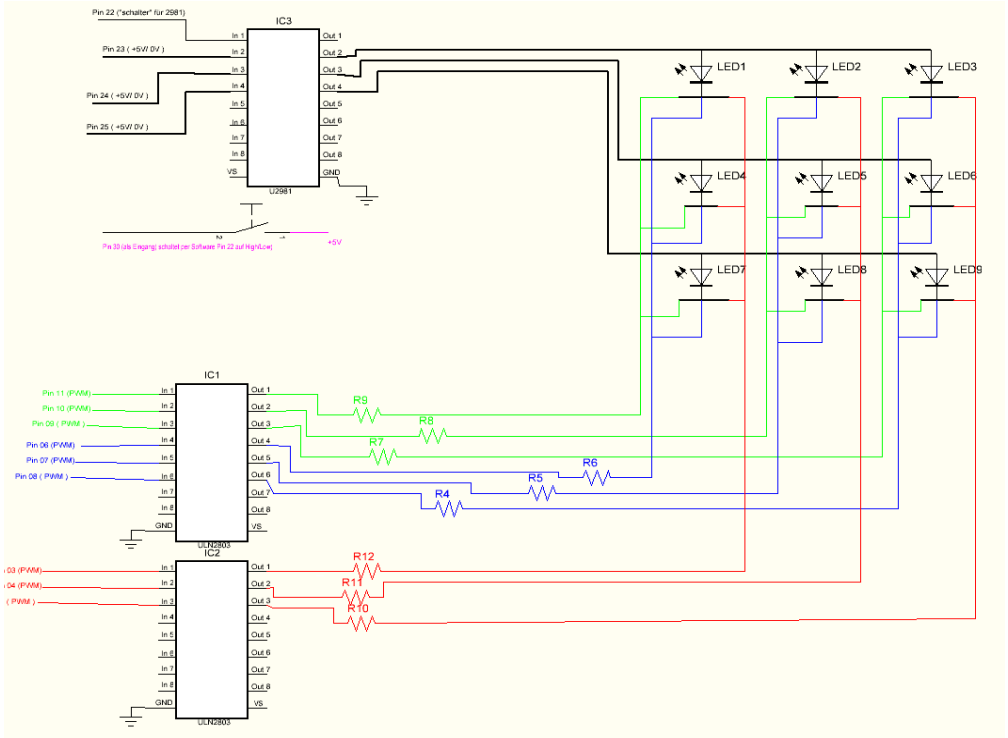
#### 5.2.3 DIE LED-MATRIX

Die LED-Matrix ist eine LED-Matrix mit 3x3 LEDs. Da wir hier eine RGB-LED Matrix haben wir für jede Farbe eine eigene Anbindung, so dass es eigentlich eine 3x3x3 Matrix ist.

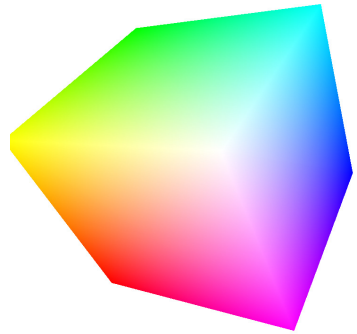
Die LEDs haben einen gemeinsamen Ausgang (Kathode) welcher zuerst in Spalte und dann gegen Spannung geschaltet wird.

Die Farben sind jeweils eine Anode pro Farbe. Weir schalten hier jeweils die einzelne Farbe der LEDs einer Reihe gemeinsam gegen einen Ausgang vom Arduino, welcher normal gegen Masse steht. Hierfür müssen an den Leitungen entsprechend Widerstände vorgeschaltet werden. So entstehen pro Reihe 3 Anschlüsse, eine für jede Farbe.

Die Anschlüsse sind jeweils über Treiber mit den Ausgängen vom Arduino verbunden.



# LIGHTBOY
## EINE LED-LAMPE



### 5.3 UMFANG DES PROTOTYPEN

An&Aus-Schalter, Default-Werte für Darstellungsmodi, Tic-Tac-Toe-Spiel,

Lichtanzeige für diverse Modus:
- Farbe für ganze Matrix
- Farbe für jede Reihe einzeln
- Farbe für jede LED einzeln

## 6 EVALUATION

Unser Ziel für die Evaluation ist die Benutzerfreundlichkeit und Nutzbarkeit des Interfaces

### 6.1 ZIELE DER EVALUATION UND METHODIK

Die Entwicklung der Lampe sollte dazu führen, dass ich mir eine eigene tolle Lampe bauen kann. Zudem wollte ich die Pläne bereitstellen, so daß andere sich auch eine gute Lampe bauen können.
Dafür ist auch wichtig, daß das Interface gut funktioniert und man damit leicht die Lampe bedienen kann.

Um dies herauszufinden haben wir ein paar Testpersonen die Lampe+Interface ausprobieren lassen.

### 6.2 DURCHFÜHRUNG DER EVALUATION

Sämtliche Tester hatten Spaß mit der Lampe. Es fiel ihnen etwas schwer sich die LED-Matrix nicht als die Test-Matrix auf dem Breadboard zu sehen, sondern wie es in Zukunft sein soll als diffuser Lichtpanell zu sehen.  Aufgrund der klarer Sicht beanstatten auch ein paar, dass nicht der gewählte Farbwert genau angezeigt wurde, sondern ein naher Farbeton, welcher dann aber flackerte.

Allerdings haben sich alle der Tester sehr stark mit dem TicTac-Toe-Modus auseinander gesetzt und lange damit gespielt.

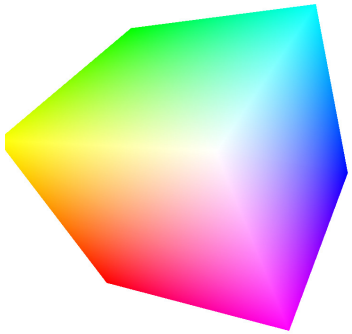
### 6.3 INTERPRETATION DER EVALUATIONSERGEBNISSE

Die Tester fanden die Lampe als gute Idee und würden gern die Weiterentwicklung des Prototypen sehen.
Vor allem würden sie sich die Steuerung und Lampe besser vorstellen, wenn man für die LED-Matrix den diffusen Lich-tkasten hat, welchen man dann auch live im Raum beobachten kann.

Auf technischer Ebene wurde bemängelt das der Button für das aktivieren des Wechseln des Darstellungsmodus zuviel sei und es ungewohnt ist den immer betätigen muss. Zudem wünschten sich 2 der Tester eine alternative die Farben zu setzen. Anstatt as man die LED und dann die Farbe auswählt, dass man die Farbe auf die LED per drag&drop ziehen kann und so setzt.

Im Grunde wurden einige Sachen genannt welche man am In-terface vereinfachen soll

# LIGHTBOY
## EINE LED-LAMPE



## 7 ZUSAMMENFASSUNG

### 7.1 ERZIELTE ERGEBNISSE

In der vorlegenden Studienarbeit wurde eine Entwicklung einer LED-Lampe und die Realisation einer Steuerung.

Der Aufbau des Interface als HTML/CSS-Anwendung stellte sich als einfach heraus. Auch das Programmieren des Arduino war gut zu realisieren. Einzig der Part welcher sich mit der Übertragung der Farbwerte vom Server zum Arduino hat sich ein wenig schwieriger herausgestellt, konnte aber mit Hilfe des Start&Endzeichens der Zeichenkette geklärt werden und die Übertragung zuverlässig gemacht werden.
Allerdings stellte sich beim Versuch heraus, dass man um die Lampe zu realisieren lieber auf ein WiFi-Shield oder Ethernet-Shield setzen sollte. So könnte man auch die Kommunikation vom HTML zum Arduino vereinfachen.
Leider konnte man mit den LED auch nicht alle Farben zuverlässig wählen, da aufgrund des PWM signals man keine richtige abstufung des Farbanteil sichern konnte und es stattdessen eher zu Flackern der LED kam, anstatt zu ner Farbveränderung.

Auch zeigt sich, dass man für das Testen schon den Lichtkasten bastelt, da die Menschen sich das Gerät als Testversion auf nem Breadbord schlecht vorstellen können.

### 7.2 ERWEITERUNGSMÖGLICHKEITEN/AUSBLICK

Bezüglich des Interfaces gibt es nur die Einführung des alternativen Farbübertragungsmodus (das Drag&Drog) und Vereinfachung des Moduswechsels kaum Verbesserungspunkte.

Hauptsächliche Erweiterungsmöglichkeiten wären das Vergrößern der LEDs auf eine 16x16 Matrix und die Umsetzung dieser als diffusen Lichtkasten. Hierfür müsste die Schaltung um die LED-Treiber TLC5940 erweitern. Durch diese könnte man a. die nötigen Ausgänge am Arduino verringern (So dass man mehrere Lichtkästen an ein Arduino schalten kann) und b. das Problem mit dem Flackern lösen und so man könnte die Farben besser abstufen.
Zudem böte sich an eine eigene Energiequelle für die Lampe mehr als an.

Aber der wichtigste Erweiterungspunkt ist erstmal ein Ethernet-Shields oder am besten WiFi-Shield einzubauen und so die Kommunikation von iPad zu Lampe zu vereinfachen und zu realisieren, damit kann man deutlich den Nutzen der Lampe erhöhen.

## 8 DANKSAGUNG

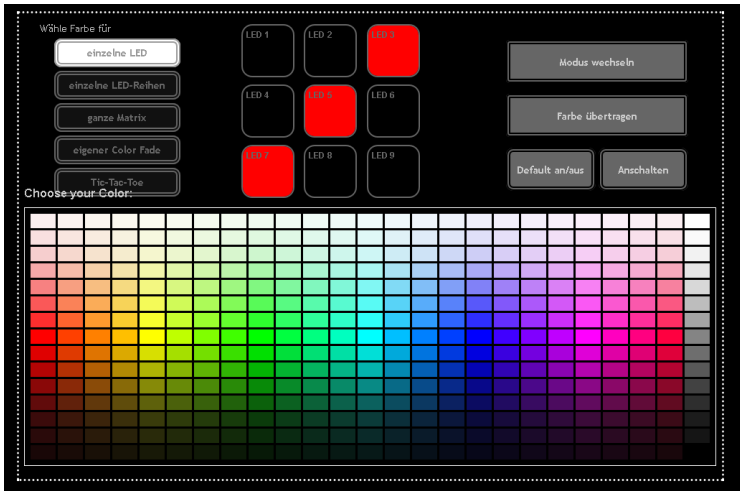
Ich danke Prof. Thomas Rist von der HS Augsburg für den positiven Feedback und Tips zu der Anwendung.

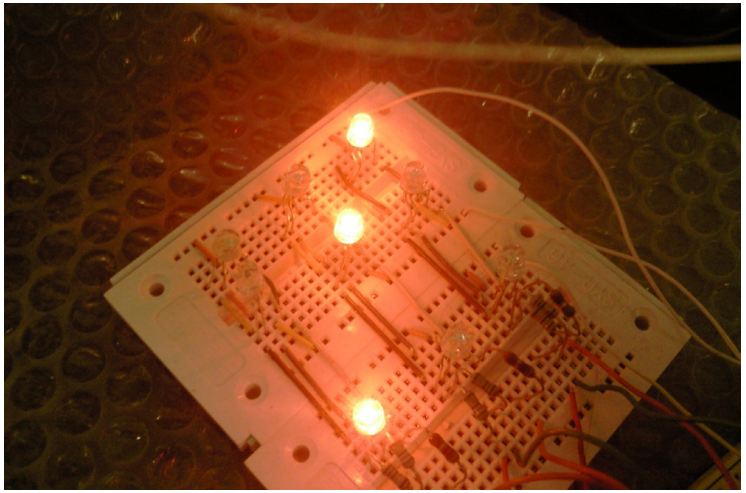
## 9 REFERENZEN

http://code.activestate.com/recipes/276962-php-tic-tac-toe/
http://arduino.cc/
http://arduino.cc/forum/index.php/board,31.0.html
http://www.instructables.com/id/Lampduino-an-8x8-RGB-Floor-Lamp/#step18

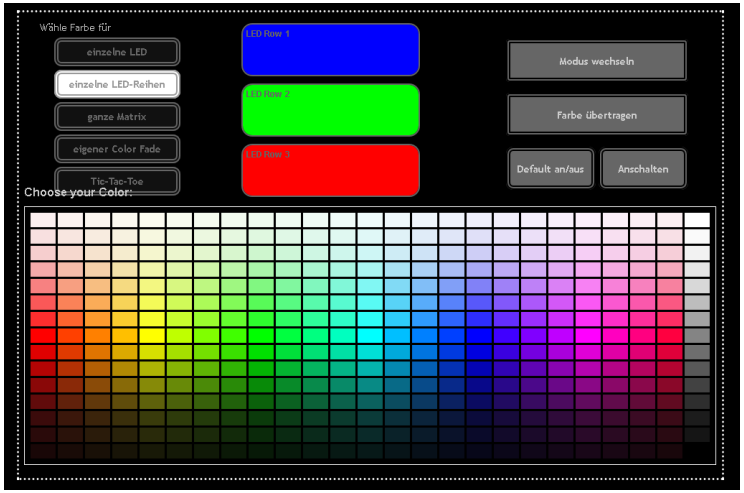
# LIGHTBOY
## EINE LED-LAMPE

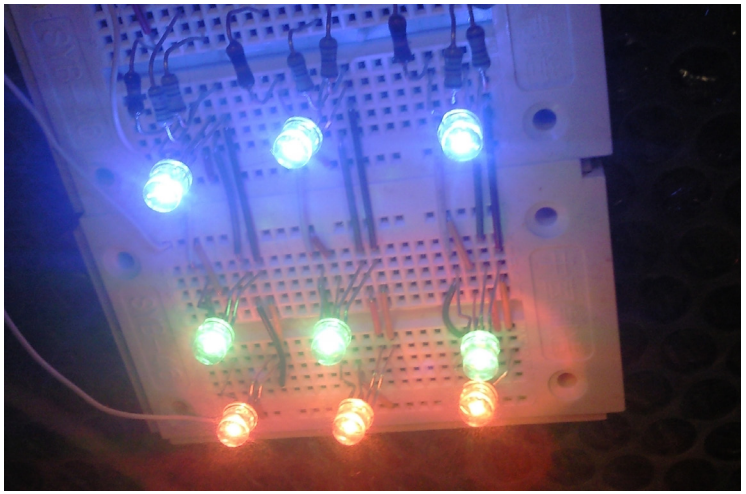
# ANHANG BILDER VOM BETRIEB



ANWENDUNGSSZENARIO 3.1 - EINZELNE LED ANSTEUERN

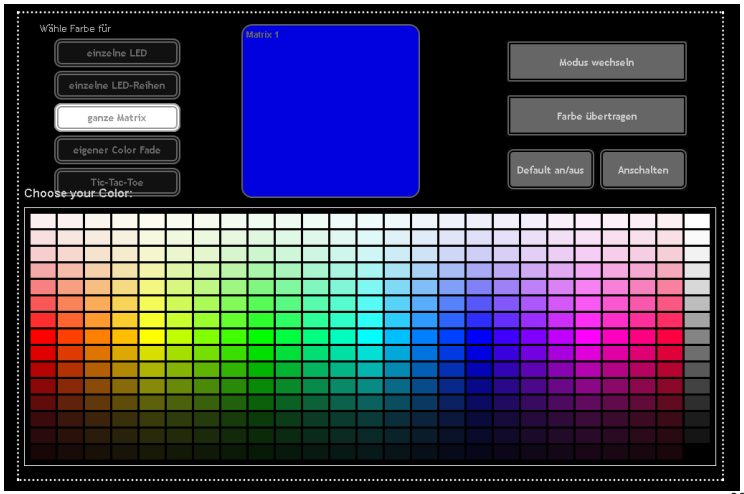


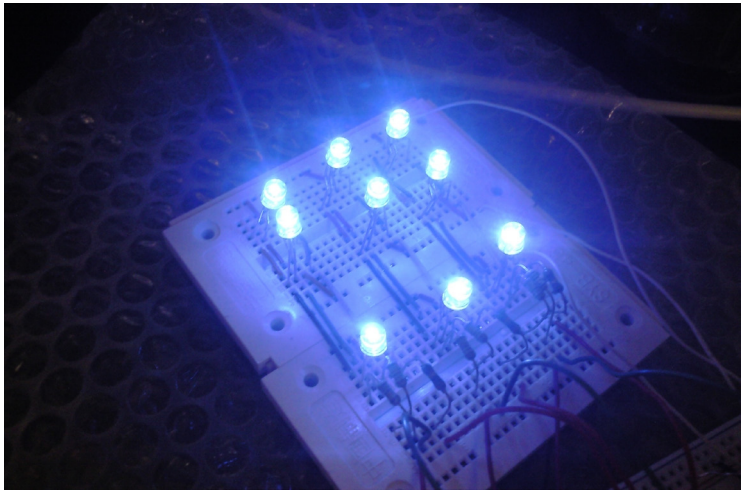



ANWENDUNGSSZENARIO 3.1 - FARBE PRO REIHE SCHALTEN

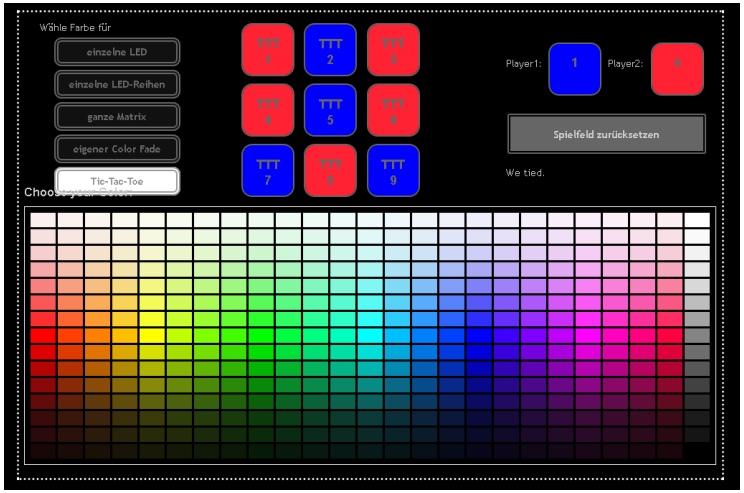
# LIGHTBOY
## EINE LED-LAMPE

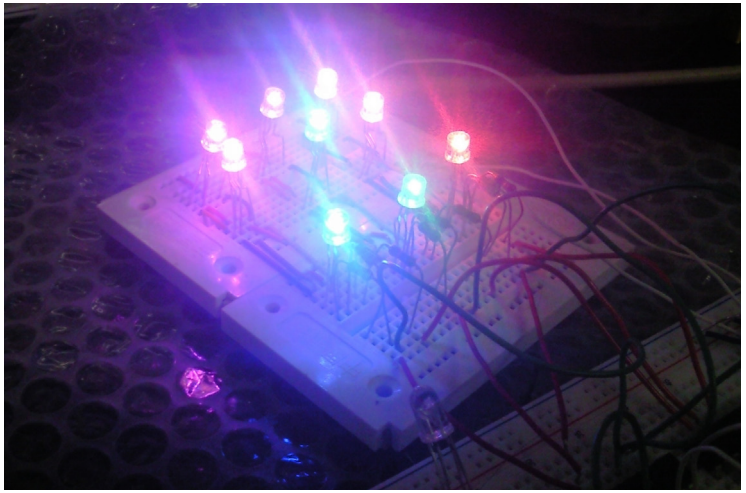
# ANHANG: BILDER VOM BETRIEB



ANWENDUNGSSZENARIO 3.1 - EINE FARBE FÜR GESAMTE MATRIX



ANWENDUNGSSZENARIO 3.4 - TICTACTOE MODUS