|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  | |  |  |
|  | | Technisches Projekt 2018/19  Medientechnik |  |
|  | |  |  |
|  | Casio Electronic Keyboard SA 77 Arduino  -  77 | |  |
|  | | Gruppe 07 / Küstennebel3000 |  |
|  | |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Name: | Dept.: | Datum: | Unterschrift: |
|  |  |  |  |
| Lennard Jansson | MT | 09.05.2019 | \_\_\_\_\_\_\_\_\_\_\_\_ |
| Nina Ozan | MT | 09.05.2019 | \_\_\_\_\_\_\_\_\_\_\_\_ |
| Mary Eufinger | MT | 09.05.2019 | \_\_\_\_\_\_\_\_\_\_\_\_ |
| Tom Haimerl | MT | 09.05.2019 | \_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

Änderungsverzeichnis

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Änderungsdatum | Geänderte Seiten | Kurzbeschreibung der Änderungen |
| 01 | 09.05.2019 | - | Erste Ausgabe |
|  |  |  |  |
|  |  |  |  |

Verteilerliste

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Firma | Name | Dept. | Funktion | Anzahl Kopien |
| HAW DMI |  | MT | Projekt Teams „Technisches Projekt“ | EMIL 🖳 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

🖳 in elektronischer Form

Inhaltsverzeichnis

1 Allgemeine Information 5

1.1 Zweck des Dokuments 5

1.2 Referenzierte Dokumente/ Online-Quellen 5

2 Systemarchitektur 6

2.1 Umsetzung der technischen Anforderungen 6

2.2 Design 6

3 Aufbau des ersten Prototyps 8

4 Detailbeschreibung „Küstennebel 3000“ 9

4.1 Hardware-Umsetzung 9

4.1.1 Stückliste 10

4.1.2 Konstruktionsplan 11

4.1.3 Schaltpläne 12

4.2 Hardware-Umsetzung 15

a) Tonausgabe am Keyboard 15

b) MIDI Steuerung 17

c) LED-Steuerung 19

d) Nebelmaschine 20

4.3 Arduino-Programmcode 21

a) Tonausgabe 21

b) MIDI Steuerung 28

c) LED-Steuerung 29

d) Nebelmaschine 31

5 Bedienungsanleitung 33

6 Anhang 36

7 QUELLEN 42

Abbildungsverzeichnis

Abbildung ‎2.2‑‎2‑1, Sensor Gehäuse 7

Abbildung 3, 1. Aufbau mit Shiftregister & Infrarot-Sensor 8

Abbildung 4.1.2, Allgemeiner Konstruktionsplan 11

Abbildung 4.1.3-1, Schaltplan Nebelmaschine 12

Abbildung 4.1.3-2, Steckplan Nebelmaschine 13

Abbildung ‎4.1.3-3, MIDI-Schaltplan 14

Abbildung ‎4.1.3-4, MIDI Steckplan 14

Abbildung 4.2a1, Diode Matrix Sound 15

Abbildung 4.2a2, Soundchip Ansteuerung 15

Abbildung 4.2a3, Lötarbeiten 16

Abbildung 4.2b1, MIDI-Buchse, eingebaut 17

Abbildung 4.2b2, MIDI-Buchse, Einbau 17

Abbildung 4.2b3, MIDI, Innenansicht 17

Abbildung 4.2b4, Bauarbeiten 18

Abbildung 4.2b5, MIDI, Breadboard 18

Abbildung ‎4.2b6, MIDI Buchse, Pinsnummerierung 19

Abbildung 4.3-1, Allgemeine Keyboard Anleitung 34

Abbildung 4.3-2, Seitenansicht Keyboard, Anleitung 34

Abbildung ‎5‑1, Zeitplan 36

# Allgemeine Information

## Zweck des Dokuments

Der Zweck dieses Dokuments ist es, unser Projekt nachvollziehbar und anschaulich mit Text und Bild darzustellen. Grundlage hierfür bilden die uns gegebenen Anforderungen der Dozenten.

## Referenzierte Dokumente/ Online-Quellen

/1/ Technisches Projekt 2017/18, Technische Anforderungen Binärrechner 1000, Version 03, 10.01.2018

Minimalanforderung

Im Rahmen des Technischen Projekts soll basierend auf einem Arduino Mikrocontroller-Board einem gegebenen elektronischen Keyboard vom Typ Casio SA-77 einem vorhandenen Lichtpult eine automatische, programmierbare Musik-/Lichtshow realisiert werden. Dabei soll der Arduino die Soundausgabe des Casio-Keyboards elektronisch ansteuern (über den integrierten Audio-Chip HT37B90) − die manuelle Eingabe über die Keyboard-Tasten soll aber weiterhin funktionieren. Der Arduino soll zu dem einige definierte MIDI-Noten ausgeben können − diese sollen dann als Input für das Lichtpult dienen, welches abhängig von der eingelesenen MIDI-Note unterschiedliche (auf die Musik abgestimmte) Lichtstimmungen erzeugen kann.  
  
Mögliche Erweiterungen

◦ Einbindung eines oder mehrerer Sensoren

◦ Spezielle Lösungen zur Bedienung der Lichtshow (z.B. über Touchscreen)

# Systemarchitektur

Die zugrundeliegenden technischen Anforderungen für das Technische Projekt „Küstennebel 3000“ mit dem Casio Keyboard und dem Ardduino sind in 1. zu finden.

## Umsetzung der technischen Anforderungen

Die minimalsten technischen Anforderungen konnten alle umgesetzt werden.

Die Matrix des Keyboards wurde ausgelesen und über den Arduino Mikrokontroller angesteuert. So kann ein Lied über das Casio Keyboard ausgegeben und abgespielt werden, während die Klaviatur weiterhin benutzt werden kann. Der Casio interne Soundchip ist mit dem Arduino-Board verdrahtet. Auf diesem befindet sich der dafür programmierte Code, siehe 4.2 a) und 4.2.1 a). Die Soundausgabe erfolgt durch die im Keyboard eingebauten Lautsprecher Boxen.

Als weitere Ausgabe ist eine MIDI-Schnittstelle verbaut. Der Ausgang wurde fest am Keyboard angebracht und mit dem Kontroller-Board verschaltet. Durch die eingebaute Buchse kann eine Kabel-Verbindung mit einem Lichtpullt aufgebaut werden und überträgt an solches, die programmierten Lichtstimmungen in MIDI Noten. Detaillierte Beschreibung der Hard- und Software der Steuerung sind unter 4.2 b) und 4.2.1 b) nachzulesen. 5 MIDI Noten (10, 20, 30, 40 und 50) und deren Lichtstimmungen wurden in Absprache mit der Lichtdepartment vorprogrammiert und zur Auswahl gestellt.

Als Erweiterungen wurden von uns zwei Show-Effekte entworfen und verbaut - eine Nebelmaschine und ein LED-Streifen.

Ein HC-SR04 Ultraschallsensor wurde eingebunden. Dieser sitzt seitlich am Casio Keyboard und löst die Nebelmaschine aus, wenn sich Hände zum Musik machen der Klaviatur nähern.

Des Weiteren wurde ein LED-Band am Keyboard befestig. Hierfür ist eine Lichtshow auf dem Arduino Board programmiert. Verschiedene Lichtstimmungen werden hier abgespielt.

Die Kosten haben das vorgesehene Budget nicht überschritten, siehe Stückliste 4.1.1.

## Design

Das Hauptdesign ist durch das Casio Keyboard vorgegeben und wurde auch bewusst nicht verändert. Die Aufmerksamkeit wird so auf unsere Features gelenkt.

Den Arduino Mega 2560 Rev3 haben wir in das Gehäuse des Keyboards eingebaut. Für die Stromversorgung des LED-Streifen, Arduino Boards und des Keybaords selbst ist ein Netzteil in Verwendung, so konnten wir mit weniger Bohrungen auskommen und das Keyboard wurde nur dezent verändert. Es gibt an der Rückseite eine MIDI-Buchse und ein Ausgang für den Strom.

Der LED-Streifen umspielt die Form des Keyboars und ist an den Kanten fest verklebt. Da das Band extra schmal und nur 1cm Breite misst, ist es optisch sehr unauffällig und bricht den gesamten Eindruck des original Casio Keyboards kaum. Erst wenn es in Betrieb ist und seine verschiedenen Leuchtmodies durchspielt, kommt es richtig zur Geltung und erzeugt einen überraschenden Effekt. Genau damit arbeiten wir auch bei der Nebelmaschine. Da die Nebelmaschine nicht direkt mit dem Keyboard in Verbindung gebracht wird. Sie ist quasi nicht sichtbar. Der Sensor sitzt in einem Gehäuse, so ist der Sensor und die davon ausgehenden Kabel zusätzlich gebündelt und geschützt. Das Gehäuse versteckt den Sensor und gibt ihm eine optische schönere Gestalt. Wobei wir uns auch zu nutzten gemacht haben, das so der Sensor am Keyboard befestigt werden kann. Das Gehäuse des Sensors fällt als markant auf, sticht raus und lässt die BenutzerInnen Ahnungslos was passieren wird. Auch lässt sich nicht eindeutig für Außenstehende erkennen, dass es sich dabei überhaupt um einen Sensor handelt. Sie Abbildung 2.2-2-1, Sensor Gehäuse



Abbildung .2‑‑, Sensor Gehäuse

# Aufbau des ersten Prototyps

Abbildung 3, 1.Aufbau mit Shiftregister & Infrarot-Sensor

# Detailbeschreibung „Küstennebel 3000“

In diesem Kapitel soll die Umsetzung des Projektes auf Hardware- sowie auf Softwareebene detaillierter betrachtet werden. Zunächst werden die Hardware-Komponenten genauer erklärt, während im weiteren Verlauf des Kapitels der Aufbau der Software gezeigt wird.

## Hardware-Umsetzung

Das Herzstück des technischen Aufbaus bildet ein Arduino MEGA 2560 mit einem ATmega2560 Prozessor. Mit seinen 54 digitalen input/output-Pins bietet er genügend Anschlussmöglichkeiten, um elektrische Signale auszusenden, die von einem Casio SA77 Keyboard in Töne umgewandelt werden können. Der Arduino ist in diesem Casio Keyboard fest verbaut, sodass er kein eigenes Gehäuse benötigt. Das Casio Keyboard ist mit seinen zwei integrierten Lautsprechern in der Lage Töne selbstständig auszugeben. Der Arduino ist mit dem Casio Keyboard über ein Flachbandkabel und einen Pfostenstecker an den Port A (digitale Pins 22 – 29) und den Port C (digitale Pins 30 – 37) des ATmega2560 verbunden. Über diese Pins ist der Arduino in der Lage mit den Soundchip kommunizieren. Die Kommunikation kann jedoch nur einseitig stattfinden: der Arduino ist in der Lage Tonsignale zu senden, kann jedoch nicht empfangen welcher Ton gerade (beispielsweise über das Drücken der Keyboardtasten) gespielt wird. Um einen Stromfluss in Richtung des Arduinos zu verhindern verläuft der Kontakt zum Keyboard über Dioden, die einen Stromfluss in diese Richtung sperren.

Der Arduino ist zum Zweiten mit einem digitalen WS2812B RGB-LED-Streifen verbunden, der um das Gehäuse des Casio Keyboard angeklebt wurde. Die einzelnen LEDs des Streifens sind einzeln ansteuerbar, sodass mit der Kombination aus Mikrocontroller und LED-Streifen zusätzliche Lichteffekte erstellt werden können. Der LED-Streifen wird über ein modifiziertes Netzteil aus einem vergangenen technischen Projekt mit 5V versorgt. Zur Steuerung der LED benötigt der Arduino lediglich einen Pin, welcher über einen 470 Ω-Widerstand zur Steuerstrombegrenzung verbunden ist. Zwischen 5V und Masse ist ein 1000μF Kondensator parallelgeschaltet, um Spannungsspitzen auszugleichen.

Die MIDI-Buchse konnte durch eine relativ simple Schaltung mit dem Arduino verbunden werden. Dieser ist in der Lage mithilfe einer vorgefertigten MIDI-Bibliothek auf Kommando im Programmcode MIDI-Noten auszugeben. Da die MIDI-Buche mit 5V arbeitet, konnten die Pins der Buchse direkt mit den 5V- und den Ground-Pins verbunden werden, wobei ein 220 Ω-Widerstand zwischen Buchse und Pin gelötet wurde. Die Datenübertragung erfolgt über den TX-Pin des Arduinos.

### Stückliste

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Anzahl | Stück | Beschreibung | Eigenschaft | Preis |
| 1  1 | Mikrocontroller  Mikrocontroller | Arduino MEGA2560  Arduino UNO |  | \*  \* |
| 1 | Kondensator |  | 1000μF | \* |
| 1  2  1  1  1  1 | Widerstand  Widerstand  Ultraschallsensor  Keyboard  Sicherung  RGB-Streifen | HC-SR04  CASIO SA-77  ?  WS2812B | 470 Ω  220 Ω  ?  1 m | \*  \*  \*  \*  \*  ca. 30 € |
| 1  1  1  1  1  1  1  1  2  1  1 | Flachbandkabel  Pfostenstecker  Nebelmaschine  Verkleidung, Sensor  Relais 833H-1C-C  Transistors  Diode  Perfboard  Buchsengehäuse-Kabel  Batterieanschluss  Batterie  Schrumpfschlauch  Draht | AF-40 Mini Fogmachine  Single pole double throw / flux tight  NPN Silicon BC337-25  1N4003-E3/54  Kupfer gekleidet  2er PTR | Holz  9V | \*  \*  Privat  \*  \*  \*  \*  Privat  \*  \*  Privat |

\* : Kostenlos erhalten aus EAW-Labor

### Allgemeiner SchaltplanKonstruktionsplan

Abbildung 4.1., Allgemeiner Konstruktionsplan

### Schaltpläne

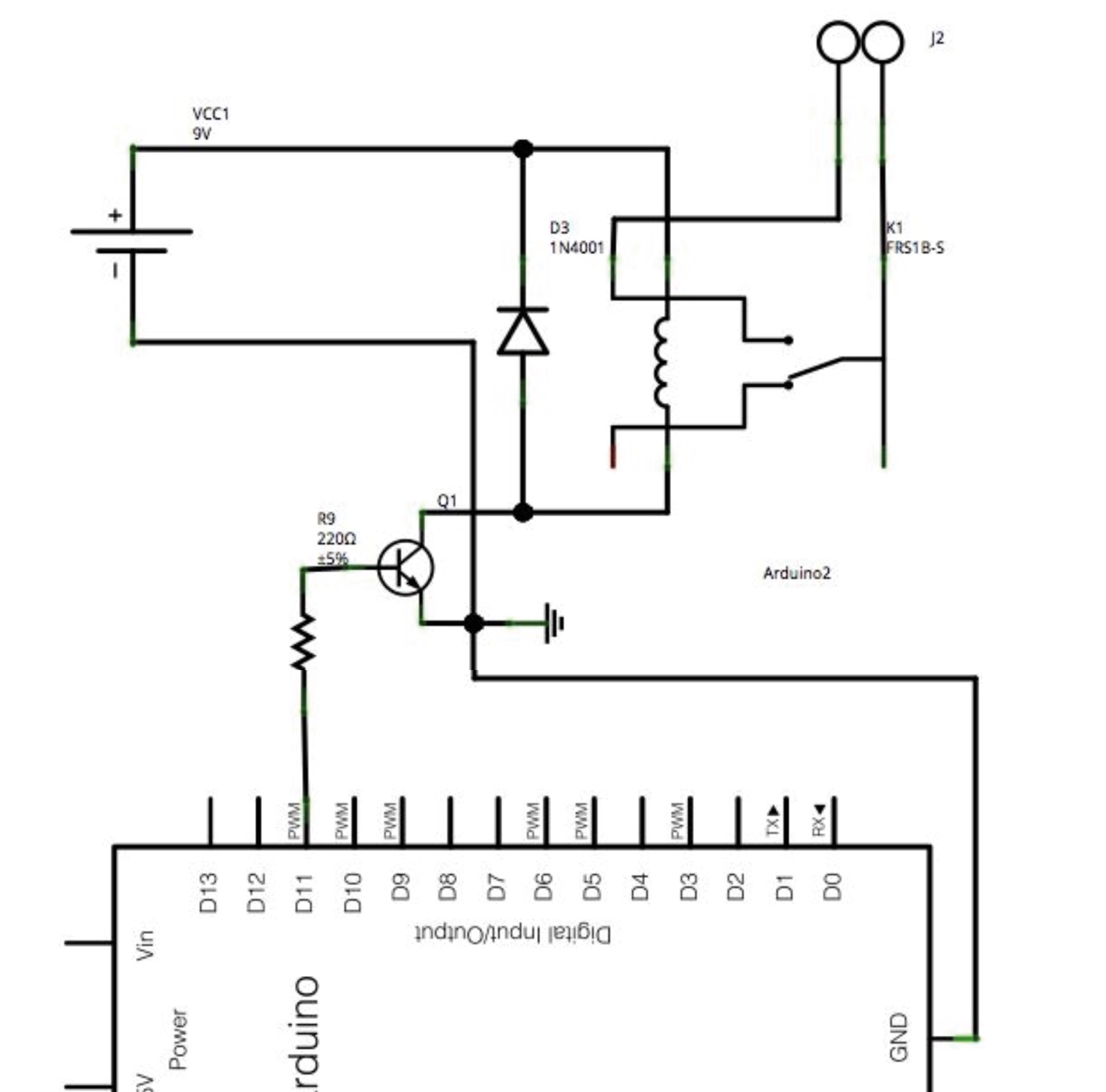


Abbildung 4.1.3-1, Schaltplan Nebelmaschine

Ein Bild, das Elektronik enthält.

Automatisch generierte Beschreibung

Abbildung 4.1.3 -2, Steckplan Nebelmaschine

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Abbildung .1.3-3, MIDI-Schaltplan

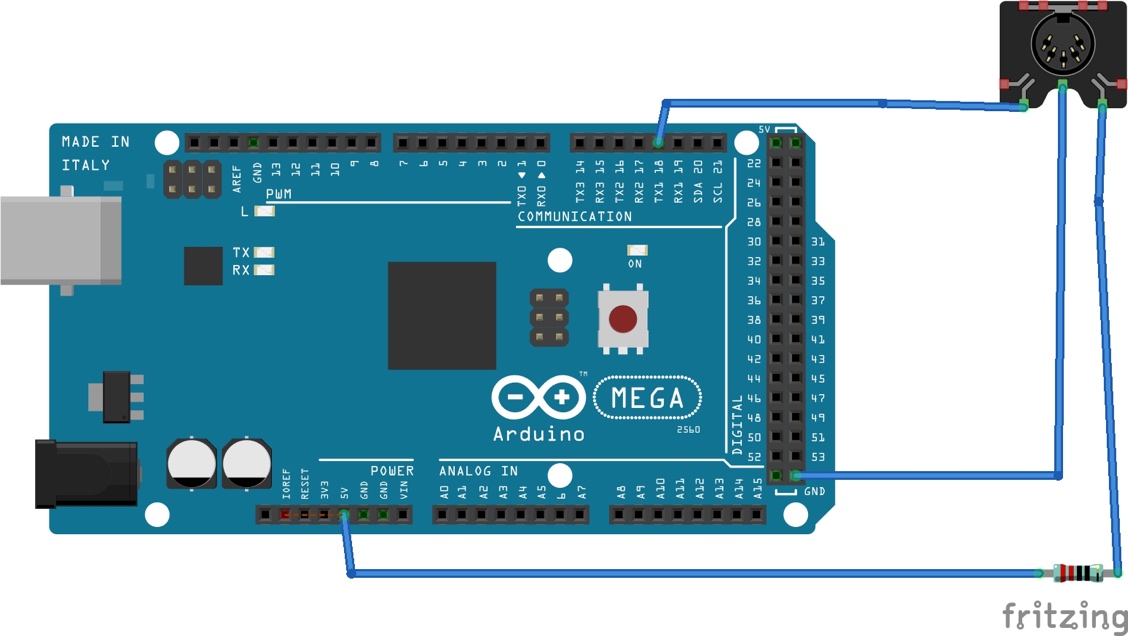


Abbildung .1.3-4, MIDI Steckplan

## Hardware-Umsetzung

### a) Tonausgabe am Keyboard

Ein Bild, das drinnen, Person enthält.

Automatisch generierte Beschreibung

Abbildung 4.2a1, Diode Matrix Sound

Ein Bild, das Elektronik, Schaltkreis enthält.

Automatisch generierte Beschreibung

Abbildung 4.2a2, Soundchip Ansteuerung

Ein Bild, das drinnen, Tisch, Person, Wand enthält.

Automatisch generierte Beschreibung

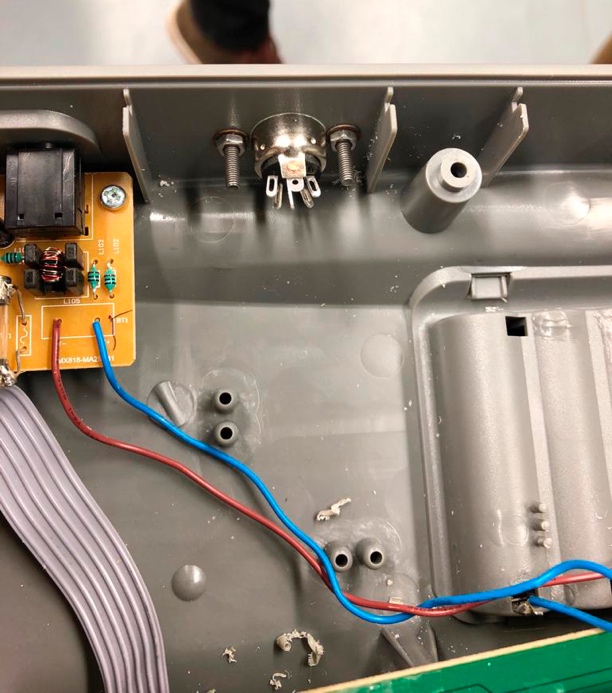
Abbildung 4.2a3, Lötarbeiten

### MIDI Steuerung

Ein Bild, das drinnen, sitzend, Elektronik enthält.

Automatisch generierte Beschreibung

Abbildung 4.2b1 MIDI-Buchse, eingebaut

Ein Bild, das Person, drinnen, Wand, Boden enthält.

Automatisch generierte Beschreibung

|  |  |
| --- | --- |
| Abbildung 4.2b2, MIDI-Buchse, Einbau | Abbildung 4.2b3, MIDI, Innenansicht |

Ein Bild, das Person, drinnen, Mann, Tisch enthält.

Automatisch generierte Beschreibung

Abbildung 4.2b4, Bauarbeiten

Ein Bild, das Elektronik, Schaltkreis enthält.

Automatisch generierte Beschreibung

Abbildung 4.2b5, MIDI, Breadboard

Es ist notwendig, ein MIDI-Interface zu verwenden, um MIDI-Befehle von einem Arduino an einen Synthesizer, eine Digital Audio Workstation oder wie in unserem Fall, an das Lichtpult zu senden. Die Schaltung der MIDI-Ausgabe gestaltet sich relativ einfach und kostengünstig. Wir haben sie permanente auf einer Leiterplatte verlötet, bzw. einen Stecker verbaut, falls der Arduino aus dem Keyboard ausgebaut werden muss. Diese Platte sitzt auf dem Arduino auf.  
Wie in Abbildung xyz, dem Schaltplan dokumentiert, ist die Schaltung für die MIDI-Ausgabe relativ einfach und besteht aus:

- eine Verbindung vom 5V-Pin des Arduinos, über einen 220 Ohm Widerstand zu Pin 4 der Standard-MIDI-DIN-Buchse

- eine Verbindung vom GND-Pin des Arduinos zu Pin 2 der MIDI-DIN-Buchse

- eine Verbindung vom TX-Pin des Arduinos zu Pin 5 der MIDI-DIN-Buchse

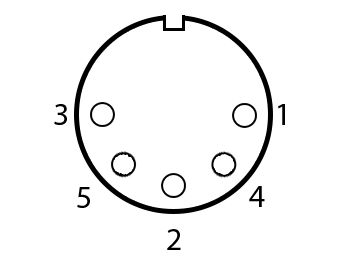


Abbildung ‎4.2b6, MIDI Buchse, Pinsnummerierung

### c) LED-Steuerung

Die Ansteuerung der RGB-LED wurde durch die Einbindung der *Adafruit NeoPixel* Bibliothek in den Arduino-Programmcode ermöglicht. Die Bibliothek stellt verschiedene Methoden bereit, um jede einzelne LED ansteuern zu können. Einige Methoden sollen im Folgenden einmal erläutert werden:

*pixels.setPixelColor(i, pixels.Color(red,green,blue));*

Setzt den Farbwert des Elements *i* auf *pixels.Color(red,green,blue)*, wobei für *red, green* und *blue* ganzzahlige Werte im Bereich von 0 – 255 eingesetzt werden können (255 für maximale Helligkeit und 0 für keine Helligkeit). Zusammen mit *pixels.Color()* ergibt das zweite Argument einen RGB-Wert bei dem für die entsprechenden Farben Werte eingesetzt werden können.

*pixels.show();*

Diese Methode lässt die LED in der durch *pixels.setPixelColor()* definierten Farbe leuchten. Das Aufrufen der Methode *pixels.setPixelColor()* alleine lässt die LEDs noch nicht in der definierten Weise leuchten.

*strip.setPixelColor() / strip.show();*

Funktionieren analog zu den oben genannten Methoden mit pixels im Namen.

*strip.numPixels();*

Gibt die im Header definierte Anzahl von LED-Elementen im verwendeten Streifen zurück.

*Adafruit\_Neopixel pixels= Adafruit(60, 8, NEO\_GRB + NEO\_KHZ800);*

Dieser Header-Befehl setzt die verwendete Anzahl von LED-Elementen im Streifen auf 60 und definiert den Arduino-Pin 8 als Daten-Pin, der an die LEDs geschlossen wird. Das letzte Argument wird für gerätespezifische Eigenschaften des verwendeten LED-Streifens verwendet.

### d) Nebelmaschine

Die Nebelmaschine soll 5 Sekunde lang Nebel ausgeben sobald sich Hände auf dem Keyboard des CASIO SA-77 Synthesizers befinden. Die Präsenzerkennung erfolgt per Ultraschallsensor HC-SR04. Die Steuerung von einem Relais ermöglicht das An -und Ausschalten der Nebelmaschine, diese Steuerung vom Relais im Verbindung zu dem Sensor erfolgt mit der Programmierung eines Arduino Uno Mikrocontrollers.

Das Ultraschallsensor HC-SR04 ist innerhalb von 2cm bis 400cm effektiv. Bei der Nebelmaschinensteuerung wird in Bereich von 2cm bis 52cm gearbeitet, dieses Bereich entspricht die 50cm Breite von der CASIO SA-77 Klaviatur.

Das Sensor ist seitlich von der Klaviatur positioniert in einem Holzgehäuse. Das Gehäuse ist in die Richtung vom Klaviatur auf und nach außen geschlossen.

## Arduino-Programmcode

### a) Tonausgabe am Keyboard

long unterschied = 0;

long interval = 1000;

long start = 0;

int takt = 1846; //LÃ¤nge des Takts festlegen. BPM von Clocks 130. -> 1846 ms Taktlänge

int resetT = 30;

byte rows = 0;

int ges = 0;

//Nummerierung für spalten und Reihen vereinfachen

byte sr1 = B01111111;

byte sr2 = B10111111;

byte sr3 = B11011111;

byte sr4 = B11101111;

byte sr5 = B11110111;

byte sr6 = B11111011;

byte sr7 = B11111101;

byte sr8 = B11111110;

byte reihen[8] = {sr1, sr2, sr3, sr4, sr5, sr6, sr7, sr8};

void setup() {

DDRC = B00000000; // Direction Register für Port C. Acht Nullen bedeuten, dass alle Pins auf Input gesetzt werden.

DDRA = B11111111; // Entsprechend für port A. Alle Pins werden auf Output gesetzt.

}

void loop() {

clocks();

//lied();

//testen();

}

void ton(int reihe, int spalte, int val){

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(reihen[reihe],reihen[spalte]);

}

}

//Dienst um einen Ton auszugeben mit den Ãœbergabeparametern reihe und spalte

void spielen(byte reihe, byte spalte) {

if (PINC == reihe) { // PINC fragt den Zustand der auf Input gesetzten Pins am Port C ab.

PORTA = spalte; // PORTA setzt den Zustand der auf Output gesetzten Pins am Port A auf 5V oder 0.

} else {

PORTA = B11111111;

}

}

//Dienst um eine bestimmte Note mit bestimmter Reihe und Spalte auszugeben mit dem Ãœbergabeparameter fÃ¼r die LÃ¤nge des zu spielenden Tons

void c2(int val) {

unterschied = 0; //unterschied auf 0 setzten

start = millis(); //start vor beginn der while Schleife auf die Millisekunden seit start des Sketches setzen

while(unterschied<=val) { //ÃœberprÃ¼fen ob Unterschied kleiner oder gleich der lÃ¤nge der Note

unterschied = millis()-start; //unterschied auf die millisekunden-start setzen

spielen(sr4,sr6); //mit den Reihen und Spalten der note, diese Ã¼ber den Dienst spielen aufrufen

}

}

void d2(int val) {

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr4,sr4);

}

}

void e2(int val) {

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr4,sr2);

}

}

void f3(int val) {

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr3,sr7);

}

}

void g3(int val) {

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr3,sr5);

}

}

void g2(int val) {

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr1,sr1);

}

}

void a3(int val) {

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr3,sr3);

}

}

void b3(int val) {

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr3,sr4);

}

}

// ----- Noten für das Lied Clocks ------------

void es3(int val){

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr6,sr5);

}

}

void ais3(int val){

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr3,sr4);

}

}

void cis3(int val){

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr5,sr3);

}

}

void as3(int val){

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr3,sr6);

}

}

void c3(int val){

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr3,sr2);

}

}

//Alle Spalten in allen Reihen auf 1 setzen

void set1(int val) {

unterschied = 0;

start = millis();

while(unterschied<=val) {

unterschied = millis()-start;

spielen(sr1,B11111111);

spielen(sr2,B11111111);

spielen(sr3,B11111111);

spielen(sr4,B11111111);

spielen(sr5,B11111111);

spielen(sr6,B11111111);

spielen(sr7,B11111111);

spielen(sr8,B11111111);

}

}

//void spiele3(void note1,void note2,void note3, int t) {

//ti = t/30;

//for(int i = 0; i<=ti; i++) {

// note1(10);

// note2(10);

//note3(10);

### b) MIDI Steuerung

#include <MIDI.h> // Erstellt und verbindet das MIDI-Interface mit der Standard-Hardware über die Serielle Schnittstelle

MIDI\_CREATE\_DEFAULT\_INSTANCE();

void setup()

{

MIDI.begin(MIDI\_CHANNEL\_OMNI); // Alle eingehende Nachrichten werden abgerufen

}

void loop()

{

MIDI.read();

MIDI.sendNoteOn(30, 127, 1); //MIDI-Note, Geschwindigkeit, Channel 1

MIDI.sendNoteOff(40, 127, 1);

MIDI.sendNoteOff(50, 127, 1);

}

### LED-Steuerung

#include <Adafruit\_NeoPixel.h>

#ifdef \_\_AVR\_\_

#include <avr/power.h>

#endif

#define PIN 8

#define NUMPIXELS 60

Adafruit\_NeoPixel pixels = Adafruit\_NeoPixel(60, 8, NEO\_GRB + NEO\_KHZ800);

Adafruit\_NeoPixel strip = Adafruit\_NeoPixel(60, 8, NEO\_GRB + NEO\_KHZ800);

void setup() {

pixels.begin();

strip.begin();

strip.setBrightness(255);

strip.show();

}

void loop() {

rainbowCycle(100);

}

void verfolgung(int freq, int red,int green, int blue){ //Sieht besonders cool bei freq>200 aus.

for(int i=0;i<NUMPIXELS;i++){

pixels.setPixelColor(i, pixels.Color(red,green,blue));

pixels.show();

delay(2000/freq);

}

for(int i=0;i<NUMPIXELS;i++){

// pixels.Color takes RGB values, from 0,0,0 up to 255,255,255

pixels.setPixelColor(i, pixels.Color(0,0,0)); // Moderately bright green color.

pixels.show(); // This sends the updated pixel color to the hardware.

delay(2000/freq); // Delay for a period of time (in milliseconds). }}

void glow(int red,int green,int blue){ // Hellt langsam in der durch RGB-Werte gegebenen Farbe auf und dunkelt danach langsam wieder ab.

int rot=0;

int gruen=0;

int blau=0;

int abdunkeln=0;

while(abdunkeln==0){

allepixel(rot, gruen, blau);

if((rot<red) || (gruen<green) || (blau<blue)){

if(rot<red){

rot++; }

if(gruen<green){

gruen++; }

if(blau<blue){

blau++; }}

else if((rot==red) && (gruen==green) && (blau==blue)){

abdunkeln = 1; }}

delayMicroseconds(1);

while(abdunkeln==1){

allepixel(rot,gruen,blau);

if(rot>0){

rot--; }

if(gruen>0){

gruen--; }

if(blau>0){

blau--; }

delayMicroseconds(1);

if((rot==0) && (gruen==0) && (blau==0)){

abdunkeln=0; }}}

void allepixel(int rot, int gruen, int blau){

for(int i=0;i<NUMPIXELS;i++){

pixels.setPixelColor(i, pixels.Color(rot,gruen,blau));

pixels.show(); }}

void colorWipe(uint32\_t c, uint8\_t wait) { // Füllt die Kette einer nach dem anderen mit einer Farbe auf.

for(uint16\_t i=0; i<strip.numPixels(); i++) { // Bsp.: colorWipe(strip.Color(255, 0, 0), 50);

strip.setPixelColor(i, c);

strip.show();

delay(wait); }}

void rainbow(uint8\_t wait) { // Alle Pixel zeigen gleichzeitig dieselbe wechselnde Farbe an.

uint16\_t i, j;

for(j=0; j<256; j++) {

for(i=0; i<strip.numPixels(); i++) {

strip.setPixelColor(i, Wheel((i+j) & 255));

}

strip.show();

delay(wait); }}

uint32\_t Wheel(byte WheelPos) {

WheelPos = 255 - WheelPos;

if(WheelPos < 85) {

return strip.Color(255 - WheelPos \* 3, 0, WheelPos \* 3);

}

if(WheelPos < 170) {

WheelPos -= 85;

return strip.Color(0, WheelPos \* 3, 255 - WheelPos \* 3);

}

WheelPos -= 170;

return strip.Color(WheelPos \* 3, 255 - WheelPos \* 3, 0);

}

void theaterChaseRainbow(uint8\_t wait) { //Lichter laufen im Uhrzeigersinn und verändern dabei ihre Farbe.

for (int j=0; j < 256; j++) { // cycle all 256 colors in the wheel

for (int q=0; q < 3; q++) {

for (uint16\_t i=0; i < strip.numPixels(); i=i+3) {

strip.setPixelColor(i+q, Wheel( (i+j) % 255)); //turn every third pixel on

}

strip.show();

delay(wait);

for (uint16\_t i=0; i < strip.numPixels(); i=i+3) {

strip.setPixelColor(i+q, 0); //turn every third pixel off

}}}}

void rainbowCycle(uint8\_t wait) { // Regenbogen, der gegen den Uhrzeigersinn läuft.

uint16\_t i, j;

for(j=0; j<256\*5; j++) { // 5 cycles of all colors on wheel

for(i=0; i< strip.numPixels(); i++) {

strip.setPixelColor(i, Wheel(((i \* 256 / strip.numPixels()) + j) & 255));

}

strip.show();

delay(wait);

}

}

### d) Nebelmaschine

/\*

HC-SR04 Ping distance sensor]

VCC to arduino 5v GND to arduino GND

Echo to Arduino pin 13

Trig to Arduino pin 12

Ausgang pin 11

\*/

const int trigPin=12;

const int echoPin=13;

const int nebel=11;

unsigned long previousMillis;

unsigned long currentMillis = millis();

void setup() {

Serial.begin (9600);

previousMillis=0;

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(nebel, OUTPUT);

}

void loop() {

long duration, distance;

digitalWrite(trigPin, LOW);

delay(10);

if((currentMillis - previousMillis) > 2){

digitalWrite(trigPin, HIGH);

delay(5);

previousMillis = currentMillis;

}

if((currentMillis - previousMillis) >10){

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = (duration/2)\*0.03432;

Serial.println(distance);

}

if (distance < 52 ) { // This is where the nebel On/Off happens

digitalWrite(nebel,HIGH); // When the condition is met, the nebel should begin

Serial.println("Nebelmaschine aktiviert.");

}

else {

digitalWrite(nebel,LOW);

}

if (distance >= 52 || distance <= 0){

Serial.println("Out of range");

}

else {

Serial.print(distance);

Serial.println(" cm");

delay(500);

}

}

# Bedienungsanleitung

Bedienungsanleitung „Küstennebel 3000“

WIR, CASIO UND ARDUINO IST NICHT HAFTBAR FÜR ETWAIGE SCHÄDEN (EINSCHLIESSLICH, ABER NICHT BEGRENZT AUF, SCHÄDEN DURCH GEWINNAUSFALL, GESCHÄFTSUNTERBRECHUNGEN ODER DEN VERLUST VON INFORMATIONEN), DIE SICH AUS DER BENUTZUNG ODER NICHTBENUTZBARKEIT DIESER BEDIENUNGSANLEITUNG ODER DES PRODUKTS ERGEBEN KÖNNTEN, AUCH WENN CASIO AUF DIE MÖGLICHKEIT SOLCHER SCHÄDEN HINGEWIESEN WORDEN IST.

Änderungen am Inhalt dieser Bedienungsanleitung bleiben ohne vorausgehende Ankündigung vorbehalten.

In dieser Bedienungsanleitung genannte Firmen- und Produktnamen sind eventuell eingetragene Marken anderer Firmen.

Wichtige Informationen für Eltern!

Eltern sollten sich mit den folgenden wichtigen Informationen vertraut machen, bevor sie ihre Kinder dieses Produkt benutzen lassen. Am besten sollten Kinder dieses Produkt nach unsere Modifikation nicht ohne Betretung benutzen. Auch der Aufbau empfiehlt sich nur durch uns oder von uns angewiesene Personen.

•Bevor Sie den Netzadapter zur Stromversorgung dieses Geräts, das Netzteil für die Nebelmaschine und den Arduino benutzen, kontrollieren Sie den Netzadapter bitte auf Beschädigungen. Kontrollieren Sie bitte das Netzkabel gründlich auf Bruchstellen, Schnitte, freiliegende Leiter und andere wesentliche Beschädigungen. Lassen Sie den Netzadapter auf keinen Fall von Kindern benutzen, wenn dieser wesentliche Beschädigungen aufweist.

Das Produkt ist nicht für Kinder ohne Beaufsichtigung bestimmt!

Stromversorgung:

Bitte bereiten Sie drei Haushaltsnetzdosen oder zwei Haushaltsnetzdosen und Batterien vor.

Wichtig! Stellen Sie sicher, dass das Digital-Keyboard ausgeschaltet ist, bevor Sie alle Netzadapter anschließen oder abtrennen oder die Batterien einsetzen oder entnehmen.

Das Keyboard hat einen DC 9,5 V-Anschluss und 5V für den LED-Streifen.

Wichtig!

•Bei langem Gebrauch wird der Netzadapter fühlbar warm. Dies ist normal und kein Hinweis auf ein Funktionsproblem.

•Um einem Drahtbruch vorzubeugen, sorgen Sie bitte dafür, dass keine Lasten jedweder Art auf das Netzkabel gestellt werden.

•Stecken Sie auf keinen Fall Metallobjekte, Stifte oder andere Gegenstände in die Anschlüsse des Produkts. Anderenfalls besteht Unfallgefahr.

Anschließen eines Kopfhörers oder Audiogeräts:

Schließen Sie den Kopfhörer bzw. das Audiogerät an die PHONES/OUTPUT-Buchse auf der Rückseite des Digital-Keyboards an.  
•Falls der Stecker nicht in die Buchse passt, verwenden Sie bitte einen im Handel erhältlichen Zwischenstecker

Allgemeine Anleitung

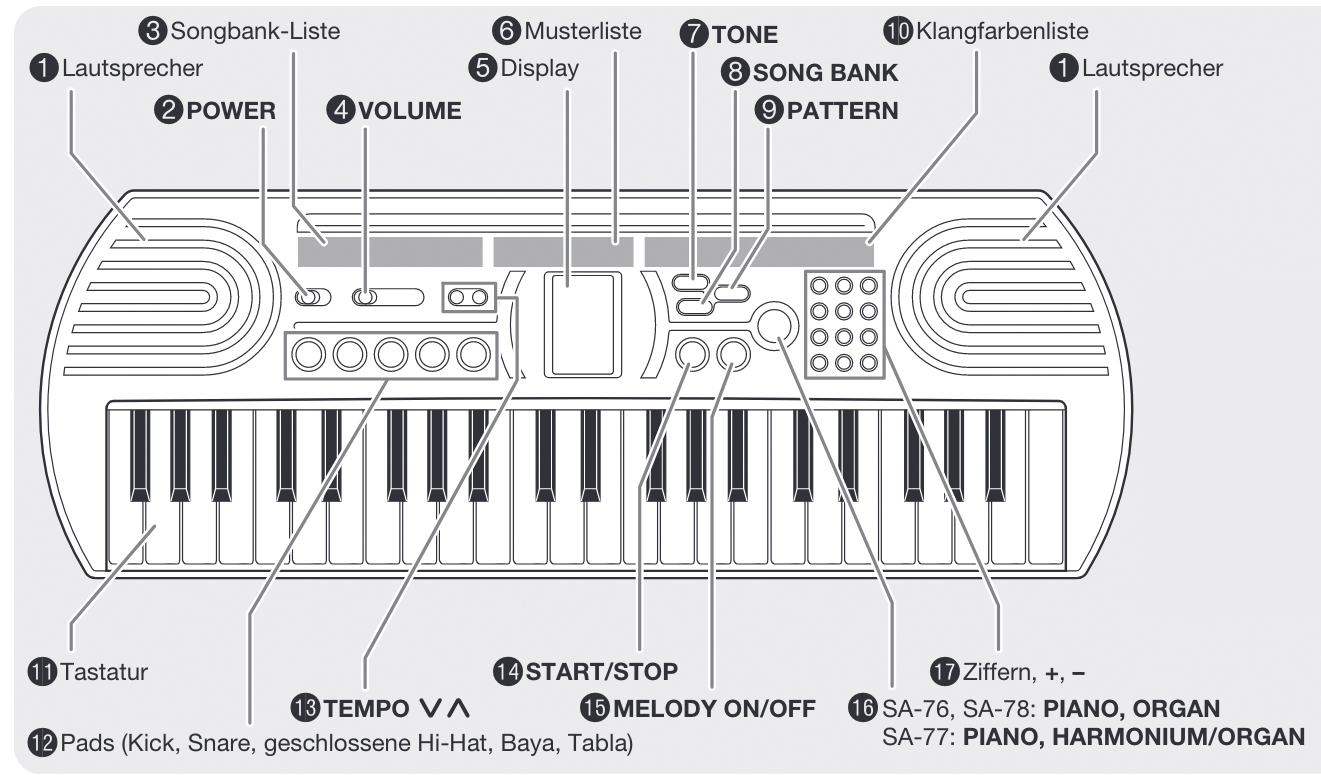


Abbildung 4.3-1 Allgemeine Keyboard Anleitung

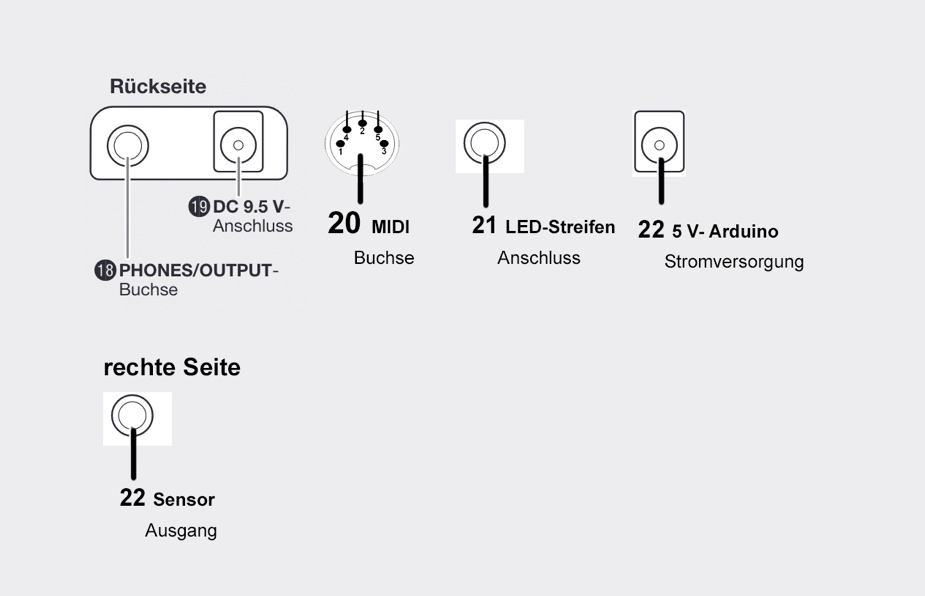


Abbildung 4.3-2 Seitenansicht Keyboard, Anleitung

Spielen auf dem Digital-Keyboard

1. Schalten Sie mit dem Schalter 2 (POWER) den Strom ein.
2. Spielen Sie versuchsweise auf der Tastatur 1I1 und den Pads 12.
3. Stellen Sie mit dem Schieberegler 4 (Volume) die Lautstärke ein.

Benutzen der vorinstallierten Effekte

1. Schließen Sie die Nebelmaschine an die Haushaltsnetzdose an.
2. Schließen Sie den Arduino 22 an die Haushaltsnetzdose an.
3. Die LEDs fangen an zu leuchten
4. Der Song wird abgespielt
5. Stellen Sie mit dem Schieberegler 4 (Volume) die Lautstärke ein.
6. Lösen Sie die Nebelmaschine beim Spielen auf der Tastatur 1I1 aus.

**Störungsbeseitigung:**

Kein Ton beim Spielen auf der Tastatur.

•Kontrollieren Sie, ob die Batterien richtig eingelegt sind oder der Netzadapter richtig angeschlossen ist. (Siehe „Stromversorgung“.)

•Ziehen Sie den Stecker aus der Buchse (9(PHONES/OUTPUT), falls diese belegt ist.

•Bei der Wiedergabe eines Songbank-Songs mit eingeschaltetem Melodie-Part ist die Tastatur (2deaktiviert. (Siehe „Wiedergeben eines Songs aus der Songbank“.)

•Dieses Digital-Keyboard ist für 8-stimmige Polyphonie geeignet. Über die achte Note hinaus erfolgt keine Wiedergabe.

**Kein Song:**

Kontrollieren Sie ob der Arduino 22 mit Strom versorgt wird.

**Kein Nebel:**

Wir kein Nebel ausgegeben, kontrollieren Sie ob genügend Nebelflüßigkeit in der Maschine ist und diese mit Strom verbunden ist.

Klang und Display verhalten sich nicht normal.

Eventuell reicht die Batterieladung nicht mehr aus. Siehe „Stromversorgung“

# Anhang

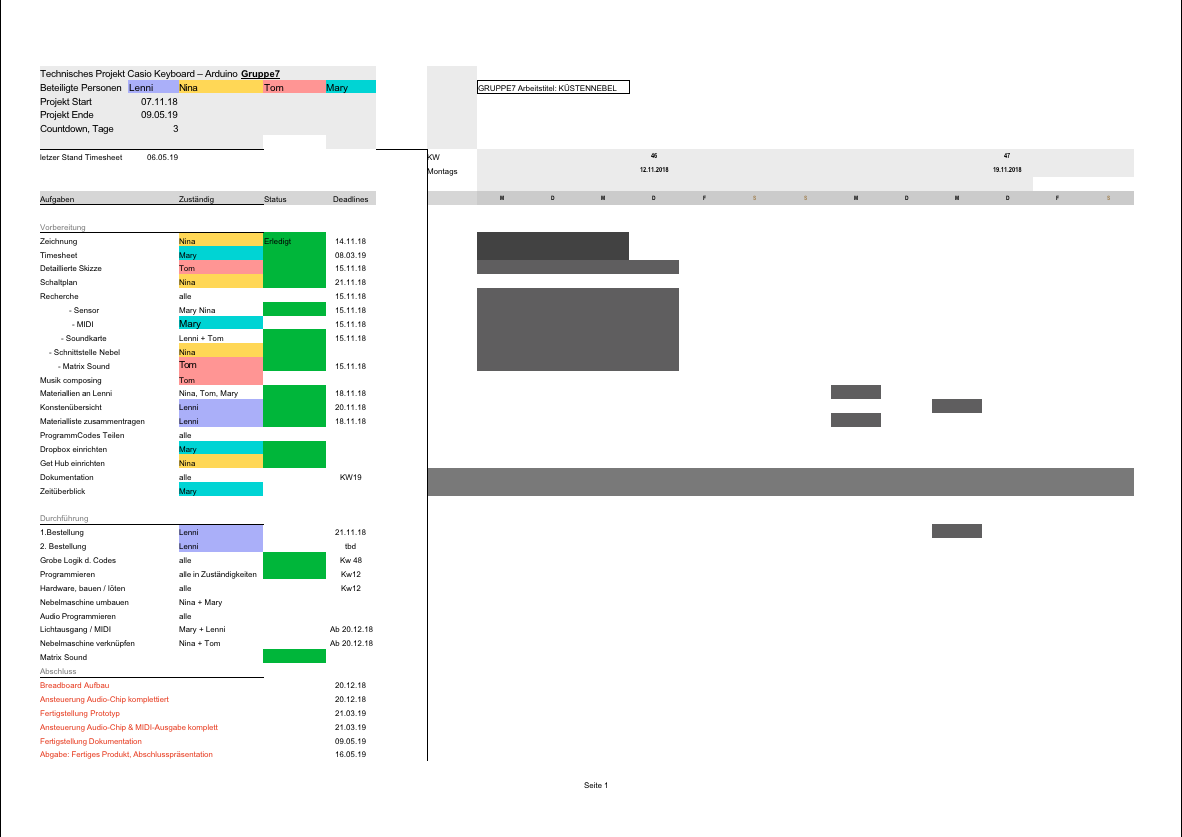


Abbildung ‑, Zeitplan



Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

# QUELLEN

4.3 Auszüge aus der Betriebsanleitung:  
https://support.casio.com/storage/de/manual/pdf/DE/008/Web\_SA76\_77\_78G1D\_DE.pdf