

Tee-Huhn-Synth

Stefanie (Momo) Beck
Wolfram Lakaszus



Der Tee-Huhn-Synth

Fotodokumentation.

Nach einem Einführungskurs in die Welt der Sensorik und der Mechatronik, wünschte sich Momo ein (zunächst) via MIDI steuerbares Objekt.

Es sollte erwartungsgemäß Töne von sich geben und zusätzlich über einen kinetischen Effekt verfügen.

Nach einer Tour durch die Dollarshops im Umfeld des Figurentheater-Kollegs in Bochum-Langendreer, lagen eine Holzschachtel und zwei Fingerpuppen auf dem Tisch.

Das Schaf und die neu erworbene Holzschachtel fielen rasch durch den Auswahlprozess:

Das konkurrierende Huhn bot deutlich mehr Sympatiepunkte als das schwarze Schaf mit seinen handlosen kurzen Armen.

Die Schachtel war langweilig und zumindest für die erste Idee, den kinetischen Effekt umzusetzen, ein wenig zu klein.



Letztere wurde schließlich durch eine hölzerne Teedose aus dem Materiallager des Medienkünstlers ersetzt.

Nach einer kurzen Phase des Experimentierens und umfangreicher mechanischer Explorationen entstand schrittweise das Konzept:

Die Schachtel sollte über ein [Standard-MIDI-Kabel](#) gesteuert werden können.

Weil das Projekt an den Tagesrändern stattfinden sollte, wurde als Grundlage ein klassischer [Arduino Uno R3](#) gewählt und es wurden keine besonderen Ansprüche an die Klangerzeugung gestellt (z.B. Verzicht auf Polyphonie, Filter und Hüllkurvengenerator).

Ausgangsbasis für den Tongenerator war ein, zur ausgewählten [MIDI-Library](#) mitgeliefertes, Beispielprogram (siehe Source-Code-Doku), das ein Rechteck-Signal mit 50% Pulsweite bei einstellbarer Frequenz erzeugt.

Bei bestimmten Noten sollte sich das Huhn aus einer Ruheposition heraus erheben. Zur mechanischen Ansteuerung wurde dazu ein kleiner Servomotor gewählt.

Im Verlaufe der Experimente entstand die zusätzliche Anforderung, dass der Tee-Huhn-Synth auch mit einer Handgeste (ähnlich dem [Theremin](#)) steuerbar sein soll. Dazu wurde ein Infrarot-Sensor, der Distanzen bis 30cm nach dem [Time-of-Flight-Prinzip](#) messen kann gewählt: [VL53L0X](#).

Der Tee-Huhn-Synth

Anwendungsformen.

Handgestensteuerung

Wie beim Theremin, erhöht sich die Tonfrequenz, je weiter der/die Spielende die Hand über dem Huhn im Tee-Huhn-Synth absenkt.

Reichweite 0 bis ca. 30cm.

Stufenlose Tonhöhe zwischen 30 und 2.000 Herz.

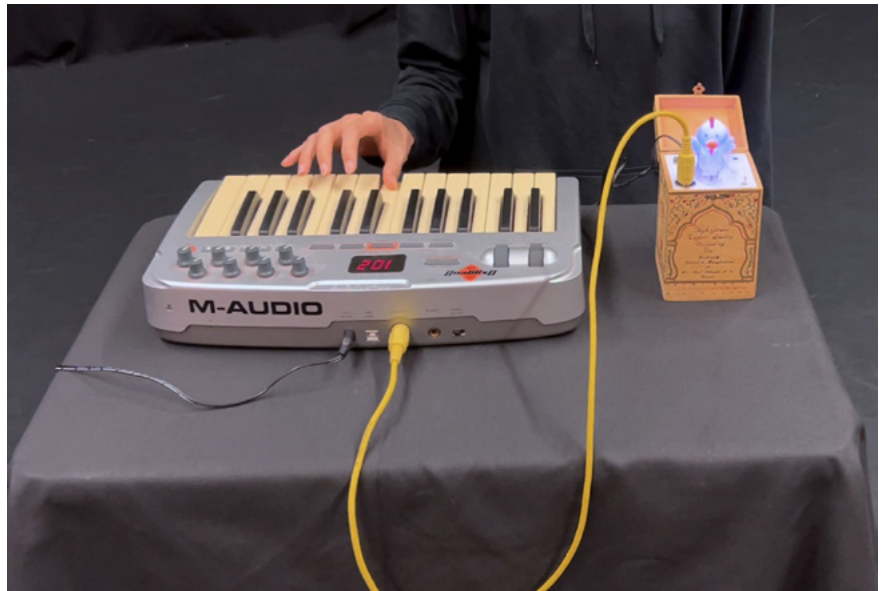


MIDI-Steuerung

Der Tee-Huhn-Synth verfügt über einen kabelgebundenen Standard-MIDI-Eingang.

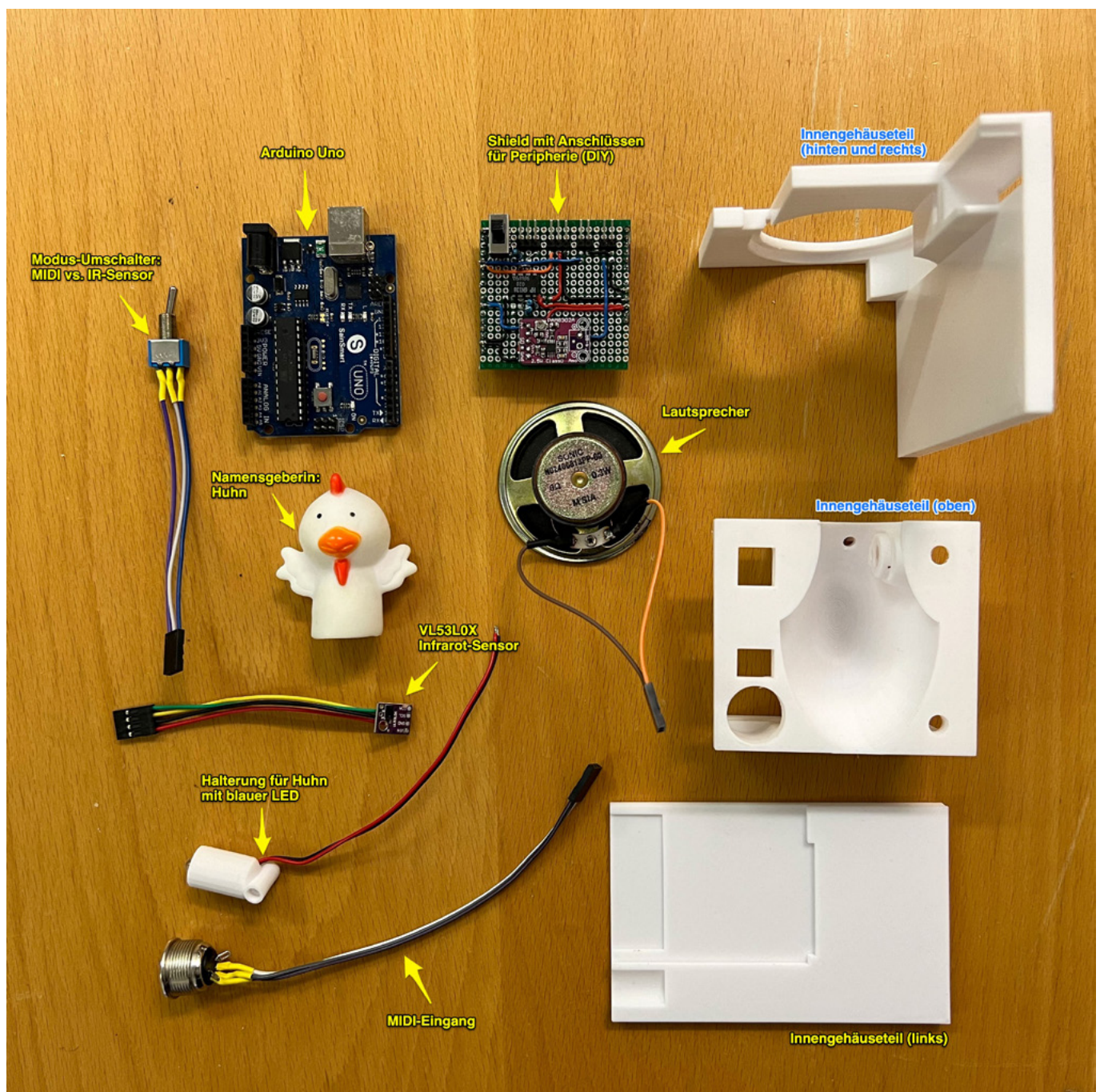
Die 128 möglichen Noten werden als Rechtecksignal erzeugt.

Wenn ein C (in allen Oktaven) angefordert wird, erhebt sich zusätzlich das Huhn des Tee-Huhn-Synth für zwei Sekunden aus seiner Ruheposition.



Hardware

Das Innengehäuse und die Elektronik.



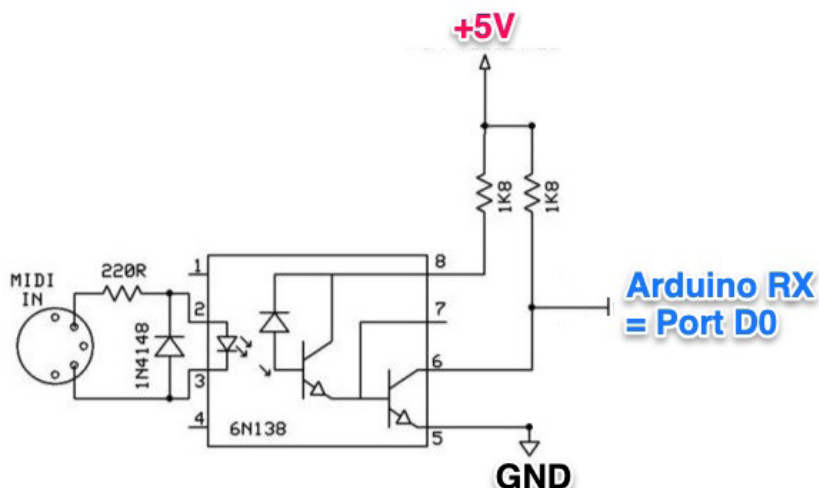
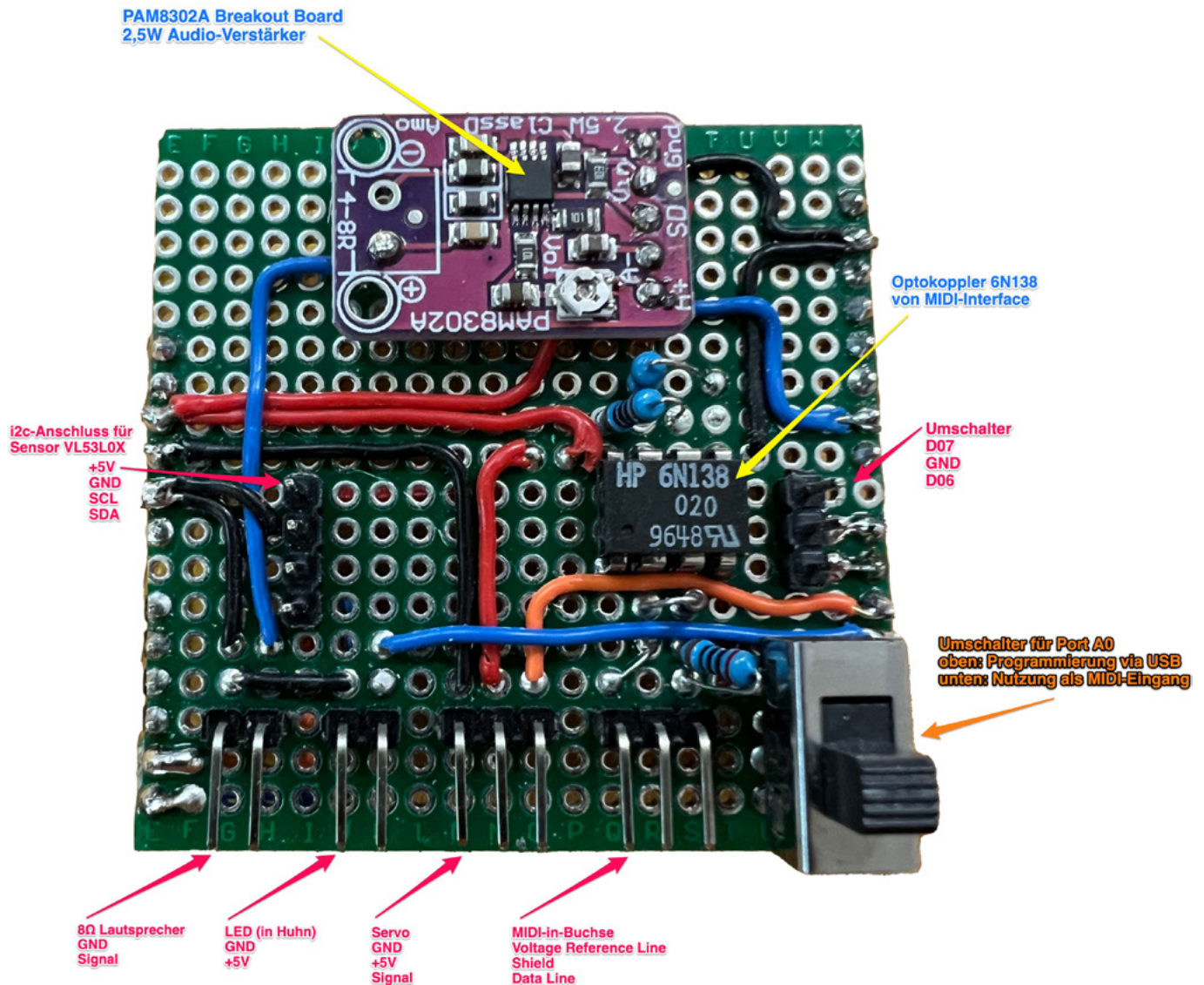
Hardware

Das Huhn, seine Halterung und seine Leuchtdiode.



Hardware

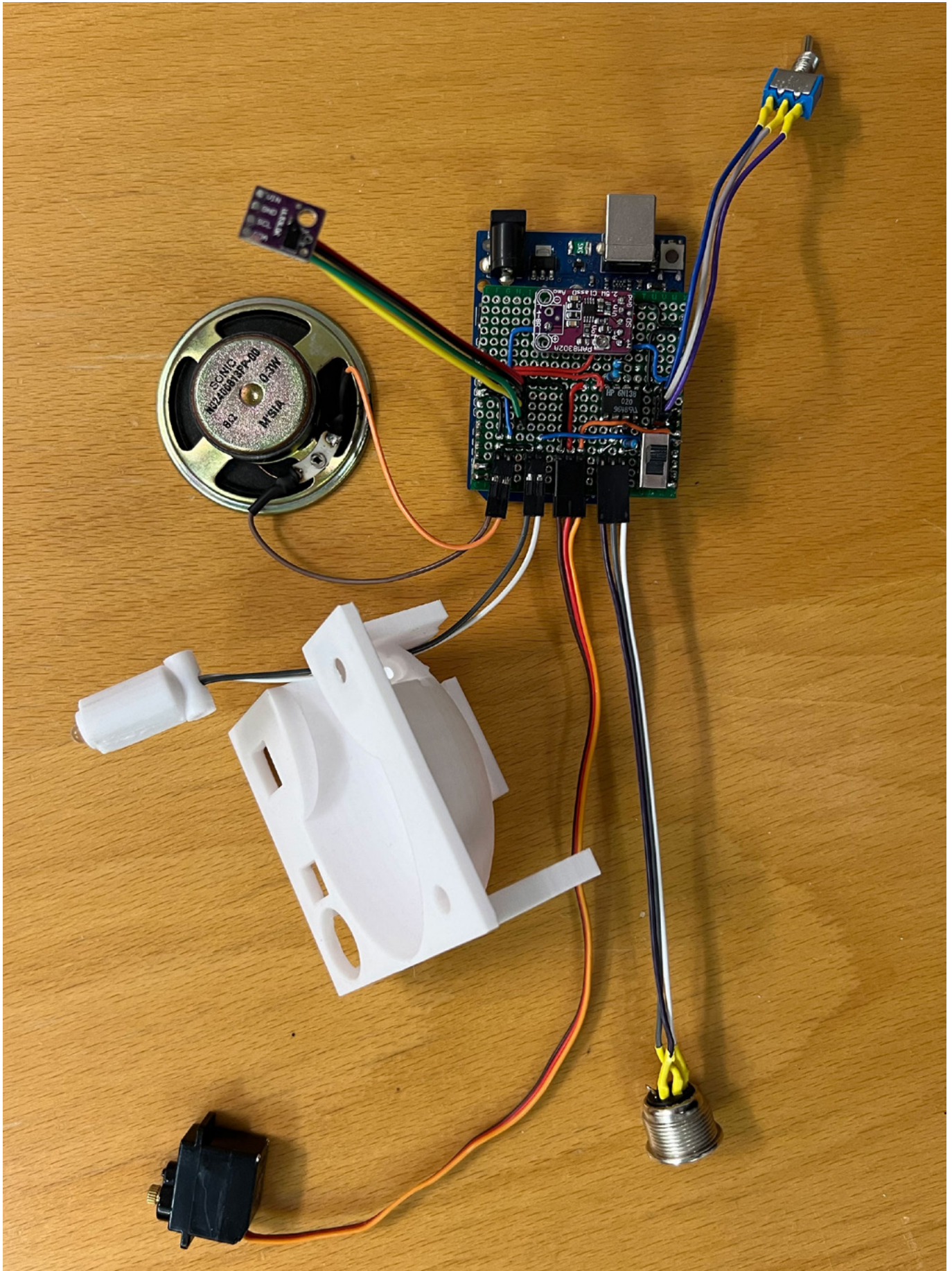
Die Pinbelegung des Shields und Schaltplan des MIDI-Interfaces.



Weiterführende Infos zum Thema MIDI-Interface selber bauen: [Arduino MIDI Interfaces – Simple DIY Electronic Music Projects](#)

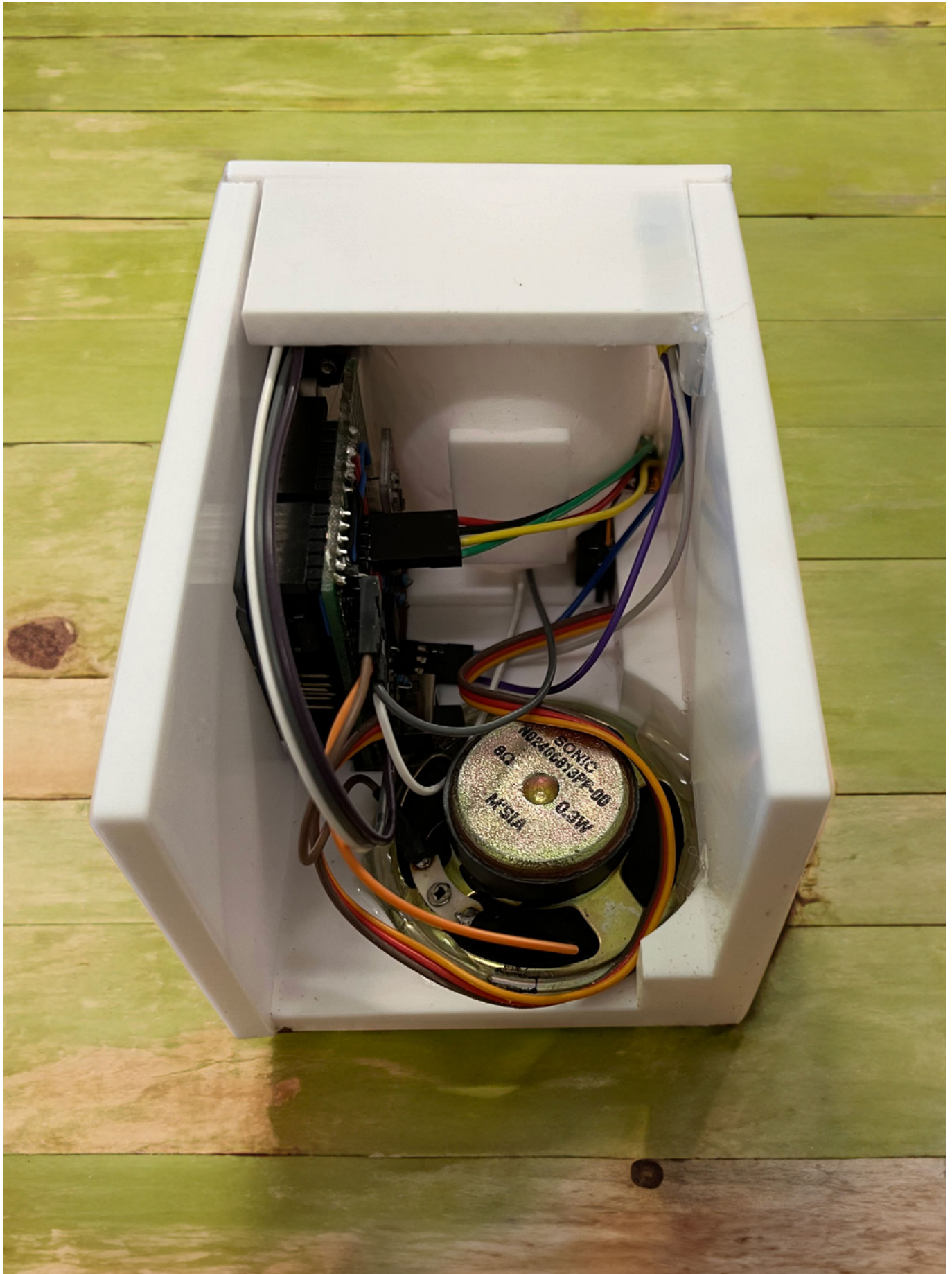
Hardware

Innereien: Das Anschlussbild der Elektronik.



Hardware

Montierte Innereien: Bereit zum Einschub in die Schachtel.



Software

Arduino-Sourcecode.

```
// -----  
//  
// Tee-Huhn-Synth  
//  
// - Base: Arduino MIDI library, SimpleSynth-Example  
//       + Servo (on/off when Note-on at Note HUHNNOTE)  
//       + VL53L0X-Sensor (Infrared time of flight distance sensor)  
//  
// - Context: Course 23-10-3436 Mechatronik: Sensoren  
//  
// - Libraries: MIDI: https://github.com/FortySevenEffects/arduino\_midi\_library  
//                  v5.0.2, Francois Best, 28.04.2020  
//                  VL53L0X: https://github.com/adafruit/Adafruit\_VL53L0X  
//                          depends: Adafruit SSD1306, Adafruit GFX Library  
//                          v1.2.3, Adafruit 22.05.2023  
//  
// Reset function (reboot) - tip from:  
// https://www.instructables.com/two-ways-to-reset-arduino-in-software  
// This function is necessary because the libraries MIDI and VL53L0X interfere with each other.  
// The Mode Switch restarts and initializes the corresponding library.  
// A better solution would be to select libraries that are compatible with each other.  
//  
// - I/O used: D0 MIDI in (with on board switch to toggle between programming and MIDI mode)  
//             D4 LED  
//             D5 Servo  
//             D6 Mode Switch  
//             D7 Mode Switch (unused)  
//             D9 Audio out (to 2.5W amplifier)  
//             D13 Analog Gate (part of demo file, unused)  
//             A4 SDA (VL53L0X-Sensor)  
//             A5 SCL (VL53L0X-Sensor)  
//  
// Select Mode via Switch at pin MODEPORT (MIDI vs. IR-Sensor)  
// Mode 0 = MIDI  
// Mode 1 = IR-Sensor  
//  
// LED at startup:  
// Fast flashing: HARDWARE ERROR - VL53L0X-Sensor not found  
// Blinking 2x 500ms: IR-Sensor mode is active  
// Blinking 1x 2s: MIDI mode is active  
//  
//  
// Stefanie (Momo) Beck, Wolfram Lakaszus  
// 28.10.2023  
// v.0.7  
// -----  
  
#include <MIDI.h>  
#include "noteList.h"  
#include "pitches.h"  
#include <Servo.h>  
#include "Adafruit_VL53L0X.h"  
  
#define HUHNPORT 5 // Port, an dem das Servo des Huhns angeschlossen ist  
#define HUHNPOSOBEN 1900 // DO NOT CHANGE! Servo-Position, wenn Huhn oben  
#define HUHNPOSUNTEN 1170 // DO NOT CHANGE! Servo-Position, wenn Huhn unten  
#define HUHNUPTIME 2000 // Zeit in ms nach der das Huhn wieder in seine Ausgangsposition geht  
#define HUHNLICHTPORT 4 // Port, an dem die LED des Huhns angeschlossen ist  
#define MODEPORT 6 // Port, an dem der Umschalter zwischen MIDI und IR-Sensor angeschlossen ist  
#define AUDIOPORT 9 // Port, an dem der Verstärker/Lautsprecher angeschlossen ist  
#define GATEPORT 13 // Port für analoges Ausgangssignal (stammt aus Demo, hier noch nicht verwendet)  
#define NOTESBUFFER 32 // Puffer für die eingehende MIDI-Events  
#define IR_TONE_MIN 30 // Lowest MIDI note, addressed by IR-sensor  
#define IR_TONE_MAX 2000 // Highest MIDI note, addressed by IR-sensor  
#define SENS_MAX 550 // VL53L0X max relevant result  
  
MIDI_CREATE_DEFAULT_INSTANCE();  
MidiNoteList<NOTESBUFFER> midiNotes;  
  
Adafruit_VL53L0X VL53L0X = Adafruit_VL53L0X();  
VL53L0X_RangingMeasurementData_t measure;  
  
Servo huhn;  
byte huhnNotes[] = {0, 12, 24, 36, 48, 60, 72, 84, 96, 108, 120}; // List of MIDI notes which will trigger the Huhn  
int huhnState = 0; // State of the Huhns (0: down, 1: up)  
long huhnUpEvent = 0; // TimeStamp, since Huhn is in up position  
int mode = 0; // 0: MIDI, 1: IR  
int oldMode = 0;  
int frequency = 0; // for tone function in IR-Sensor mode
```

```

// -----
void(* reboot) (void) = 0; // declares reset function at address 0
// Tipp: https://www.instructables.com/two-ways-to-reset-arduino-in-software

// -----
boolean isHuhnNote (int note) { // check if actual note is in list relevant notes (to trigger Huhn motion)
    boolean isRelevant = 0;
    for (int i=0; i<sizeof(huhnNotes); i++){
        if (note==huhnNotes[i]) isRelevant = 1;
    }
    return isRelevant;
}

// -----
inline void handleGateChanged(bool inGateActive) { // not yet used in this applikation
    digitalWrite(GATEPORT, inGateActive ? HIGH : LOW);
}

// -----
inline void pulseGate() { // not yet used in this applikation
    handleGateChanged(false);
    delay(1);
    handleGateChanged(true);
}

// -----
void handleNotesChanged(bool isFirstNote = false) {
    if (midiNotes.empty()) {
        handleGateChanged(false);
        noTone(AUDIOPORT); // Remove to keep oscillator running during envelope release.
    } else {
        byte currentNote = 0;
        if (midiNotes.getLast(currentNote)) {
            tone(AUDIOPORT, sNotePitches[currentNote]);
            if (isFirstNote) {
                handleGateChanged(true);
            } else {
                pulseGate(); // Retrigger envelopes. Remove for legato effect.
            }
        }
    }
}

// -----
void handleNoteOn(byte inChannel, byte inNote, byte inVelocity) {
    const bool firstNote = midiNotes.empty();
    midiNotes.add(MidiNote(inNote, inVelocity));
    digitalWrite(HUHNLICHTPORT, 1);
    if (isHuhnNote(inNote) && (huhnState == 0)) {
        huhn.writeMicroseconds(HUHNPOSOBEN); // Servo swings Huhn up
        huhnState = 1;
        huhnUpEvent = millis();
    }
    handleNotesChanged(firstNote);
}

// -----
void handleNoteOff(byte inChannel, byte inNote, byte inVelocity) {
    midiNotes.remove(inNote);
    handleNotesChanged();
    digitalWrite(HUHNLICHTPORT, 0);
}

// -----
void handleHuhnUpTime() { // check, if its time to swing Huhn back in starting position
    if ((huhnState==1) && (millis()-huhnUpEvent>HUHNUPTIME)){
        huhn.writeMicroseconds(HUHNPOSUNTEN); // Servo swings Huhn back
        huhnState = 0;
    }
}

```



```

// -----
void setup() {
  // ---- Ports ----
  pinMode(AUDIOPORT, OUTPUT);
  pinMode(HUHNLICHTPORT, OUTPUT);
  pinMode(MODEPORT, INPUT_PULLUP);
  mode = digitalRead(MODEPORT);
  oldMode = mode;

  if (mode==0) { // mode 0 = MIDI-Mode
    // ---- MIDI ----
    digitalWrite(HUHNLICHTPORT, HIGH);
    delay(2000);
    digitalWrite(HUHNLICHTPORT, LOW);
    MIDI.setHandleNoteOn(handleNoteOn);
    MIDI.setHandleNoteOff(handleNoteOff);
    MIDI.begin(); // listens on all MIDI channels. Option: put specific channel in brackets
    // ---- Servo ---- // Servo (Huhn) is only active in MIDI-Mode
    huhn.attach(HUHNPORT);
    huhn.writeMicroseconds(HUHNPOSUNTEN); // Servo gets Huhn in starting position

  } else { // mode 1 = IR-Sensor-Mode
    // ---- VL53L0X ----
    digitalWrite(HUHNLICHTPORT, HIGH);
    delay(500);
    digitalWrite(HUHNLICHTPORT, LOW);
    delay(500);
    digitalWrite(HUHNLICHTPORT, HIGH);
    delay(500);
    digitalWrite(HUHNLICHTPORT, LOW);
    if (!VL53L0X.begin()) { // failed to boot VL53L0X
      while (1) {
        digitalWrite(HUHNLICHTPORT, HIGH);
        delay(100);
        digitalWrite(HUHNLICHTPORT, LOW);
        delay(100);
      }
    }
  }
}

// -----
void handleMode() { // check if position of mode switch has changed
  mode = digitalRead(MODEPORT);
  if (mode!=oldMode){
    reboot();
  }
}

// -----
// -----
void loop() {
  handleMode(); // check if position of mode switch has changed

  if (mode==0) {
    // ---- MIDI ----
    MIDI.read(); // get MIDI input to buffer and play notes from buffer
    handleHuhnUpTime(); // check, if its time to swing Huhn back in starting position

  } else {
    // ---- VL53L0X ----
    VL53L0X.rangingTest(&measure, false); // do not pass in 'true' (to get debug data printout): MIDI uses
    // serial port!
    if (measure.RangeStatus != 4) { // phase failures have incorrect data
      int sens=measure.RangeMilliMeter; // get distance
      if (sens>0 && sens<SENSMAX){ // if result is in relevant range
        frequency = map(measure.RangeMilliMeter,0,SENSMAX,IR_TONE_MIN,IR_TONE_MAX); // map result to frequency
        tone(AUDIOPORT, frequency); // play tone(frequency)
      } else {
        noTone(AUDIOPORT); // shut up aka. stop playing tone
      }
    }
  }
}

```

Referenzen

Quellen und Credits

Ein Projekt von **Stefanie (Momo) Beck** und **Wolfram Lakaszus**
am Figurentheater-Kolleg in Folge des Kurses

23-10-3436

Mechatronik: Sensoren

Ein Einstieg in die Mechatronik mit einem Fokus auf Sensoren

Bochum, Oktober 2023



Libraries:

MIDI: [Francois Best](#)

VL53L0X: [Adafruit](#)

Reset Funktion: [Gabriella Levine](#)

Hardwarequellen (Beispiele, Stand: 28.10.2023):

Arduino Uno R3: [Original](#), kompatible Clones bei [ebay](#), bei [AliExpress](#)

VL53L0X Breakoutboard bei [ebay](#), bei [AliExpress](#)

Servomotor bei [ebay](#), bei [AliExpress](#)

PAM8302A Breakoutboard bei [ebay](#), bei [AliExpress](#)

Optokoppler 6N138 bei [Reichelt](#), [ebay](#), [AliExpress](#)

Innenrahmen:

Modell: [Tee-Huhn-Synth](#) bei [TinkerCAD](#)

Filament: [Sunlu PLA+ White](#), bei [ebay](#)

3D-Drucker: [Bambu Lab X1-Carbon Combo](#)

Fotos, Illustrationen:

Wolfram Lakaszus, Figurentheater-Kolleg

Tee-Huhn-Synth in Aktion:



<https://vimeo.com/880966233>



Figurentheater-Kolleg
Hohe Eiche 27
44892 Bochum
+49-234-284080
info@ft-k.de