Fachhochschule der Wirtschaft
FHDW

Bergisch Gladbach

Studienarbeit

Digitales Theremin mit Raspberry Pi und Leap Motion

Autor:
Vasilij Schneidermann

Prüfer:
Dr. Peter Tutt

Studiengang:
Wirtschaftsinformatik

Fachbereich:
Consulting

Abgabetermin:
07. April 2016

Inhaltsverzeichnis

[1] Open Sound Control

Codeverzeichnis

1 Einführung

Ähnlich wie der populäre Arduino-Microcontroller hat der Raspberry Pi Kleinstcomputer der breiten Öffentlichkeit näher gebracht. Mögliche Einsatzgebiete sind stromsparende Medienserver, Verwendung im Unterricht und Experimente mit dem *Internet of Things*. Die vorliegende Studienarbeit konzentriert sich auf ein etwas ungewöhnlicheres Projekt in dessen Rahmen ein digitales Theremin unter Zuhilfenahme einer Leap Motion als kontaktloses Eingabegerät umgesetzt wurde.

2 Grundlagen

2.1 Raspberry Pi

Der Raspberry Pi ist ein im Jahr 2011 eingeführter Einplatinencomputer[1]. Ziel seiner Einführung war die Verwendung im Unterricht und Entwicklungsländern.

Das verwendete Gerät weist unter anderem USB-Ports, GPIO[2]-Pins, einen Ethernet-Port, HDMI-Anschluss und einen SD-Kartenleser[3]. Letzterer wird verwendet um ein Betriebssystem, in unserem Falle Raspbian[4] zu starten.

2.2 Leap Motion

Die Leap Motion ist ein Eingabegerät welches es mithilfe von Kameras und Infrarotsensoren ermöglicht hochauflösend Handbewegungen zu verfolgen[5]. Die dabei anfallenden Daten können nach Installation des offiziellen SDK[6] mithilfe einer API[7] für eine Programmiersprache ausgewertet werden.

2.3 Python

Python ist eine populäre dynamische Programmiersprache[8]. Eines der Ziele von Python ist es Programme lesbar und leicht verständlich zu gestalten. Die Sprache unterstützt verschiedenste Programmierparadigmen, hat ein großes Ökosystem mit über $75,000$ Modulen[9] und ist insbesondere im wissenschaftlichen Umfeld beliebt.

[1] BBC (2011), Online im Internet
[2] General Purpose Input Output
[3] Raspberry Pi Foundation (2016), Online im Internet
[4] Raspbian (2016), Online im Internet
[5] Leap Motion Inc. (2016), Online im Internet
[6] Software Development Kit
[7] Application Programming Interface
[8] Python Software Foundation (2016), Online im Internet
[9] PyPi (2016), Online im Internet

2.4 ChucK

ChucK ist eine Programmiersprache für die Synthese, Komposition und Analyse von Musik[10]. Eine Besonderheit im Vergleich zu anderen Programmiersprachen in diesem Bereich ist der starke Fokus auf Zeit: Um Ton ausgeben zu können muss der Programmierer explizit Zeit verstreichen lassen. Diese ungewöhnliche Designentscheidung ermöglicht es individuelle Komponenten der Synthese frei miteinander zu synchronisieren und auf das der Situation angemessene Detaillevel herunterzugehen.

2.5 OSC

OSC ist ein im Audiobereich breit verwendetes Protokoll und Format zur Kommunikation zwischen Instrumenten[11]. Im Gegensatz zu dem populären MIDI[12]-Standard ist es deutlich leichtgewichtiger, allgemeiner und einfacher zu implementieren.

2.6 Theremin

Das Theremin ist ein elektronisches Musikinstrument welches mit den Händen berührungsfrei gespielt wird. Es ist nach seinem Erfinder, Léon Theremin benannt. Die linke Hand kontrolliert die Lautstärke, die rechte Hand die Tonhöhe.

[10] ChucK (2016), Online im Internet
[11] Open Sound Control (2016), Online im Internet
[12] Musical Instrument Digital Interface

3 Planung und Übersicht

Die anfängliche Idee die Leap Motion direkt an den Raspberry Pi anzuschließen wurde verworfen, da laut offizieller Stellungsnahme des Herstellers der Raspberry Pi nicht die nötigen Mindestanforderungen erfüllt[1]. Stattdessen wird in einem Blogpost für ein vergleichbares Projekt[2] vorgeschlagen die Leap Motion an einem leistungstärkerem Rechner zu betreiben und den Raspberry Pi die benötigten Daten periodisch abfragen zu lassen. Für die Kommunikation müssten dann ein Client und ein Server geschrieben werden welche ein gemeinsames Protokoll sprechen. Zur Verbindung von dem Rechner und Raspberry Pi wird ein Netzwerk mithilfe eines Switch und zwei LAN[3]-Kabeln aufgebaut.

Da die offizielle Anbindung an die GPIOs[4] in Python geschrieben ist und die Leap Motion auch eine Python-API[5] aufweist, wurde der Einfachheit halber Python für beide Komponenten verwendet. Als Protokoll wird HTTP[6] mit JSON[7] als *Payload* verwendet, für die Server-Seite ist Flask[8] und für die Client-Seite Requests[9] zuständig.

Das andere Problem ist die Audiosynthese. Zwar ist es möglich eine vorhandene Audiobibliothek zu nutzen und den Ton Sample für Sample zu konstruieren, jedoch wurde diese Vorgehensweise als zu aufwändig für das Ziel eingestuft[10]. Aufgrund dessen wurden stattdessen drei verschiedene Programmiersprachen für diesen Zweck entwickelte evaluiert, SuperCollider[11], Csound[12] und ChucK[13]. Super Collider war die anfangs favorisierte Lösung, wurde aber wegen mangelnder Dokumentation für konsolenbasierte Steuerung und einer Inkompatibilität auf der ARM-Architektur des Raspberry Pi schließlich verworfen. Csound sieht sehr spannend aus, kam aber wegen des Fokus auf orchestralen Kompositionen nicht in Frage. ChucK hat brauchbare Dokumentation und funktionierte mit wenig Setup auf dem Raspberry Pi und wurde deswegen ausgewählt.

[1] Leap Motion Community (2013), Online im Internet
[2] Leap Motion Blog (2015), Online im Internet
[3] Local Area Network
[4] PyPi RPi.GPIO (2016), Online im Internet
[5] Leap Motion Developer (2016), Online im Internet
[6] Hyper-Text Transfer Protocol
[7] JavaScript Object Notation
[8] Armin Ronacher (2016), Online im Internet
[9] Kenneth Reitz (2016), Online im Internet
[10] Wäre das Ziel hingegen DSP-Programmierung, würde dieser Ansatz viel mehr Sinn ergeben.
[11] SuperCollider (2016), Online im Internet
[12] Csound (2016), Online im Internet
[13] ChucK (2016), Online im Internet

Für die Modulation von Tonhöhe und Lautstärke ist ebenfalls ein Protokoll nötig. ChucK unterstützt das Empfangen von OSC-Nachrichten nativ, deswegen wird nur noch pyOSC[14] auf der Client-Seite benötigt.

Der Aufbau lässt sich in folgender Grafik zusammenfassen:

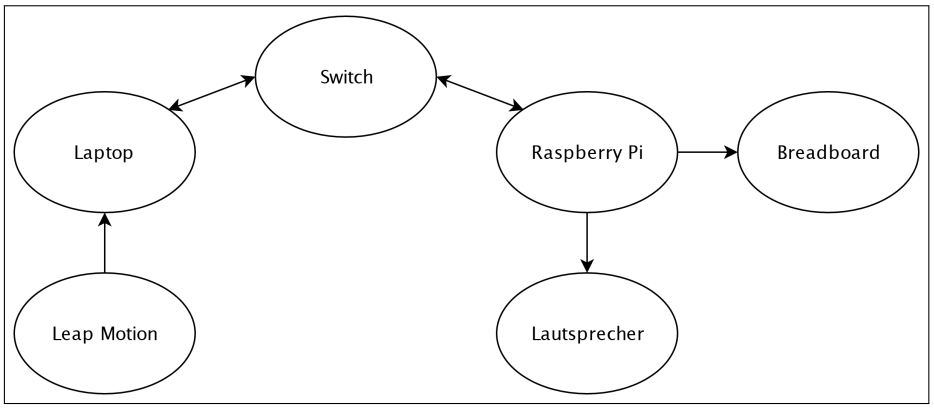

Abbildung 3.1: Digitales Theremin, Quelle: Eigene Abbildung

[14] pyOSC (2010), Online im Internet

4 Umsetzung

4.1 Netzwerksetup

Der Raspberry Pi wird ohne vorinstalliertes Betriebssystem geliefert, daher ist es notwendig zuerst ein *Image* herunterzuladen und es auf eine SD-Karte zu schreiben. Die Quellen und Anleitungen dafür werden vom Raspbian-Projekt bereitgestellt[1], einem Debian[2]-Derivat für den Raspberry Pi. Um den Computer bedienen zu können, ist es notwendig Tastatur und Monitor anzuschließen.

Das Netzwerk ist sehr einfach aufgebaut. Dem Raspberry Pi wird die IP-Adresse `192.168.178.2` und dem Laptop `192.168.178.24` zugewiesen[3]. Unter Zuhilfenahme eines DHCP[4]-Servers und der von Google bereitgestellten Adresse `8.8.8.8` für die DNS[5]-Auflösung kann die Konfiguration auf ein Minimum beschränkt werden. Fügt man Port Forwarding und Masquerading hinzu, kann der Raspberry Pi die Internetverbindung vom Laptop mitverwenden was z.B. für die Installation neuer Pakete äußerst hilfreich ist.

Die Konfiguration wird mithilfe des `ping`-Tools getestet: Es sollte möglich sein von beiden Seiten aus die andere Seite zu erreichen. Im Anschluss dessen kann man sich vom Laptop aus mit `ssh pi@192.168.178.23` auf den Raspberry Pi über SSH[6] verbinden[7].

Schnitzer bei der Konfiguration können ausgebessert werden indem man die SD-Karte herausnimmt, in einen anderen Rechner einbindet und die Datei `/boot/cmdline.txt` editiert. Um z.B. mit einer festen IP-Adresse erreichbar zu sein, genügt es den Parameter `ip=<address>` hinzuzufügen. Dieser kann anschließend nach erfolgreicher Reparatur des Netzwerksetups für einen normalen Bootvorgang wieder entfernt werden.

[1] https://www.raspbian.org/RaspbianImages
[2] https://www.debian.org/
[3] Wichtig hier ist darauf zu achten, dass diese Adressen noch nicht im Netzwerk vergeben sind oder noch besser, Teil eines anderen als des aktuell verwendeten lokalen Netzwerks sind.
[4] Dynamic Host Configuration Protocol
[5] Domain Name System
[6] Secure SHell
[7] Annahme hier ist dass der Benutzername auch `pi` ist

4.2 Handtracking

Die Leap Motion-API bietet zwei Modi an: Event-basiert (für die Verarbeitung aller Daten) und Polling (bedarfsgesteuerte Verarbeitung von Daten). Da nur ein Bruchteil der Daten periodisch angefragt wird, ist letzterer Ansatz sinnvoller. Folgender Code ist in Aufarbeitung und Bereitstellung der Daten aufgeteilt:

```
import leap
import logging
from flask import Flask, jsonify

log = logging.getLogger('werkzeug')
log.setLevel(logging.ERROR)

app = Flask(__name__)
controller = leap.Controller()

def hand_state():
    frame = controller.frame()
    hands = frame.hands
    state = {'left': None, 'right': None}

    for hand in hands:
        if hand.is_valid:
            if hand.is_left:
                state['left'] = hand.palm_position.y
            elif hand.is_right:
                state['right'] = hand.palm_position.y

    return state

@app.route('/')
def api_root():
    return jsonify(hand_state())

def main():
    app.run(host='0.0.0.0')

if __name__ == '__main__':
    main()
```

Anschließend kann der Server mithilfe von `python laptop.py` gestartet werden. In einem anderen Terminal kann man die Funktionalität dessen durch `curl http://localhost:5000` testen. Je nach Handstatus sollte JSON mit relativen Distanzen oder dem Sonderwert `null` für eine nicht vorhandene Hand zurückgegeben werden.

4.3 Visuelles Feedback

Um mehr Rückmeldung vom Raspberry Pi erhalten zu können wird ein *Breadboard* mit zwei Leuchtdioden-Schaltkreisen bestückt:

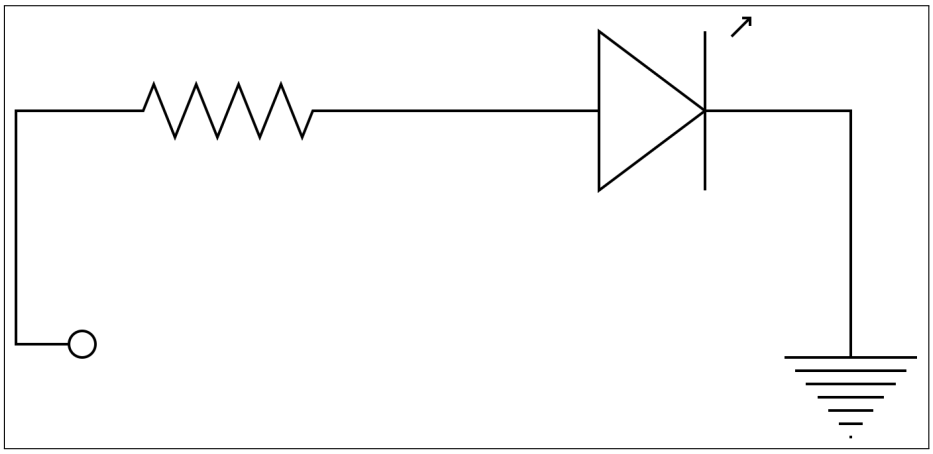

Abbildung 4.1: LED-Schaltkreis, Quelle: Eigene Abbildung

Die Idee dabei ist, dass man beim Spielen des Instruments informiert wird ob die Hand zur Steuerung der Tonhöhe sich im validen Bereich befindet oder darüber hinaus geht. Im ersteren Fall wird eine grüne Leuchtdiode angeschaltet und eine rote Leuchtdiode ausgeschaltet, ansonst gilt das gleiche umgekehrt. Der Raspberry Pi ermöglicht es mit der Mehrzahl seiner GPIO-Anschlüsse einen elektrischen Schalter zu implementieren. Folgender Code geht davon aus, dass einmal Pin 5 und 6 (BCM 3) sowie Pin 40 und 39 (BCM 21) angeschlossen sind[8]:

```
import RPi.GPIO as GPIO
import requests
import time

from OSC import OSCMessage
from OSC import OSCClient

RED_LED = 3
```

[8]Zur Lokalisierung der Pins ist http://pinout.xyz/ hilfreich

```
GREEN_LED = 21
SERVER = 'http://192.168.178.24:5000'

def led_setup():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(RED_LED, GPIO.OUT)
    GPIO.setup(GREEN_LED, GPIO.OUT)

def red_led_on():
    GPIO.output(RED_LED, 1)

def red_led_off():
    GPIO.output(RED_LED, 0)

def green_led_on():
    GPIO.output(GREEN_LED, 1)

def green_led_off():
    GPIO.output(GREEN_LED, 0)

def led_teardown():
    GPIO.cleanup()

def hand_state():
    return requests.get(SERVER).json()

def led_control(state):
    distance = state['right']
    if distance and distance > 100 and distance < 600:
        red_led_off()
        green_led_on()
    else:
        red_led_on()
        green_led_off()

def clamp(n, lower, upper):
    if n < lower:
        return 0.0
```

```
    elif n >= lower and n < upper:
        return (n - lower) / float(upper)
    else:
        return 1.0

def main():
    led_setup()
    client = OSCClient()
    address = ('127.0.0.1', 6666)
    while True:
        state = hand_state()
        led_control(state)
        if state['left']:
            msg = OSCMessage('/volume')
            msg.append(clamp(state['left'], 100, 600))
            client.sendto(msg, address)
        if state['right']:
            msg = OSCMessage('/pitch')
            msg.append(clamp(state['right'], 100, 600))
            client.sendto(msg, address)
        time.sleep(0.01)
    led_teardown()

if __name__ == '__main__':
    main()
```

Ähnlich wie das vorige Code-Beispiel wird dieses mit `python raspberry.py` ausgeführt. Funktioniert alles, kann man die grüne Leuchtdiode durch das Bewegen der rechten Hand über den Sensor zum Leuchten bringen.

Das PyOSC-Modul ist nicht direkt aus den Raspbian-Paketarchiven installierbar, deswegen ist eine manuelle Installation notwendig:

```
$ curl -Ok https://trac.v2.nl/raw-attachment/wiki/pyOSC/\
pyOSC-0.3.5b-5294.tar.gz
$ cd pyOSC-0.3.5b-5294
$ sudo ./setup.py install
```

4.4 Audiosynthese

Es ist anzumerken, dass ChucK unter Raspbian nicht aktuell genug für das nachfolgende Beispiel ist. Für die manuelle Installation sind folgende Schritte nötig:

```
$ curl -O \
  http://chuck.cs.princeton.edu/release/files/chuck-1.3.5.2.tgz
$ tar -xf chuck-1.3.5.2
$ cd chuck-1.3.5.2/src
$ sudo apt-get install gcc bison flex libasound2 \
  libasound2-dev libsndfile1 libsndfile1-dev
$ make linux-alsa
$ sudo make install
```

Bei genauerer Beobachtung eines Theremins fällt auf, dass es mehr Anforderungen als die Steuerbarkeit von Lautstärke und Tonhöhe mit der Hand gibt:

- Reine, stumpfe Klangfarbe
- Niederfrequente Modulation der Tonhöhe
- Linearisiertes Frequenzfeld
- Übergangslose Änderung der Tonhöhe

Eine Umsetzung aller Punkte findet sich in folgendem Code-Beispiel in ChucK:

```
SinOsc sin => dac;
0.5 => dac.gain;
440.0 => float oldFreq;
440.0 => float newFreq;
440.0 => sin.freq;

SinOsc lfo => blackhole;
5 => lfo.freq;

OscIn oscpitch;
6666 => oscpitch.port;
"/pitch" => oscpitch.addAddress;

OscIn oscvol;
6666 => oscvol.port;
"/volume" => oscvol.addAddress;

fun float fmin(float a, float b) {
```

```
    return (a < b) ? a : b;
}

fun void interpolate() {
    1.5 => float step;
    while (true) {
        fmin(oldFreq, newFreq) + Math.fabs(oldFreq - newFreq) /
            step => oldFreq;

        oldFreq => sin.freq;
        5::ms => now;
    }
}

fun void wobble() {
    while (true) {
        lfo.last() * 0.01 * oldFreq +=> oldFreq;
        10::ms => now;
    }
}

fun void adjustVolume() {
    while (true) {
        oscvol => now;
        while (oscvol.recv(OscMsg msg)) {
            Std.fabs(msg.getFloat(0)) => dac.gain;
        }
    }
}

spork ~ interpolate();
spork ~ wobble();
spork ~ adjustVolume();

while (true) {
    oscpitch => now;
    while (oscpitch.recv(OscMsg msg)) {
        Std.fabs(msg.getFloat(0)) => float pitch;
        Math.pow(2, pitch * 2 + 8) => newFreq;
```

```
    }
}
```

Zum Testen führt man das obige Code-Beispiel mit `chuck theremin.ck` aus. Sollte nichts zu hören sein, kann es sein, dass ChucK nicht die richtige Soundkarte erkannt hat. Man kann alle bekannten mithilfe von `chuck --probe` auflisten, ist z.B. Soundkarte 3 die erwünschte, verwendet man `chuck --dac3 theremin.ck`.

Die Klangfarbe wurde mithilfe eines Sinusoszillators angenähert. Dies entspricht nicht ganz einem authentischen Theremin[9], nähert dieses aber hinreichend an. Für die niederfrequente Modulation wird ein LFO[10] verwendet welcher die Tonhöhe mit einer Frequenz von 5 hz moduliert.

Die Linearisierung des Frequenzfeldes klingt schwierig, ist aber leicht umsetzbar wenn man bedenkt, dass eine Verdopplung der Frequenz einen Oktavensprung bedeutet. Im obigen Beispiel werden die Eingabewerte zwischen 0.0 und 1.0 auf Frequenzen zwischen $2^8 hz$ und $2^{10} hz$ verteilt.

Die Übergänge werden mithilfe einer einfachen Interpolation zwischen den Zeitpunkten jedes eingehenden OSC-Pakets geglättet. Man startet mit einer Ausgangsfrequenz, das OSC-Paket setzt eine Zielfrequenz und die Interpolation setzt die Ausgangsfrequenz auf einen Zwischenwert.

[9] Dafür wäre eine leichte Verzerrung der Wellenform notwendig, diese wurde der Einfachheit halber ausgelassen

[10] Low-Frequency Oscillator

5 Ehrenwörtliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Studienarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Vasilij Schneidermann

Bergisch Gladbach, den 7. April 2016.

6 Quellenverzeichnis

Internetquellen

[1] BBC (2011), *BBC - dot.Rory: A 15 pound computer to inspire young programmers*, Online im Internet: http://www.bbc.co.uk/blogs/thereporters/rorycellanjones/2011/05/a_15_computer_to_inspire_young.html, Stand: 07.04.2016

[2] Raspberry Pi Foundation (2016), *Raspberry Pi 2 Model B*, Online im Internet: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/, Stand: 07.04.2016

[3] Raspbian (2016), *FrontPage - Raspbian*, Online im Internet: https://www.raspbian.org/, Stand: 07.04.2016

[4] Leap Motion Inc. (2016), *Leap Motion | Mac & PC Motion Controller for Games, Design, Virtual Reality & More*, Online im Internet: https://www.leapmotion.com/, Stand: 07.04.2016

[5] Python Software Foundation (2016), *Welcome to Python.org*, Online im Internet: https://www.python.org/, Stand: 07.04.2016

[6] PyPi (2016), *PyPi - the Python Package Index : Python Package Index*, Online im Internet: https://pypi.python.org/pypi, Stand: 07.04.2016

[7] ChucK (2016), *ChucK => Strongly-timed, On-the-fly Music Programming Language*, Online im Internet: http://chuck.cs.princeton.edu/, Stand: 07.04.2016

[8] Open Sound Control (2016), *opensoundcontrol.org | an Enabling Encoding for Media Applications*, Online im Internet: http://opensoundcontrol.org/, Stand: 07.04.2016

[9] Leap Motion Community (2013), *Leap Motion support on Raspbmc (Raspberry Pi) - Customer Support - Leap Motion Community*, Online im Internet: https://community.leapmotion.com/t/leap-motion-support-on-raspbmc-raspberry-pi/155, Stand: 07.04.2016

[10] Leap Motion Blog (2015), *How to Integrate Leap Motion with Arduino & Raspberry Pi*, Online im Internet: http://blog.leapmotion.com/integrate-leap-motion-arduino-raspberry-pi/, Stand: 07.04.2016

[11] PyPi RPi.GPIO (2016), *RPi.GPIO 0.6.2 : Python Package Index*, Online im Internet: https://pypi.python.org/pypi/RPi.GPIO, Stand: 07.04.2016

[12] Leap Motion Developer (2016), *Python SDK Documentation - Leap Motion Python SDK v2.3 documentation*, Online im Internet: https://developer.leapmotion.com/documentation/python/index.html, Stand: 07.04.2016

[13] Armin Ronacher (2016), *Welcome | Flask (A Python Microframework)*, Online im Internet: http://flask.pocoo.org/, Stand: 07.04.2016

[14] Kenneth Reitz (2016), *Requests: HTTP for Humans - Requests 2.9.1 documentation*, Online im Internet: http://docs.python-requests.org/en/master/, Stand: 07.04.2016

[15] SuperCollider (2016), *SuperCollider » SuperCollider*, Online im Internet: https://supercollider.github.io/, Stand: 07.04.2016

[16] Csound (2016), *Home | Csound Community*, Online im Internet: https://csound.github.io/, Stand: 07.04.2016

[17] pyOSC (2010), *pyOSC - V2_Lab Projects*, Online im Internet: https://trac.v2.nl/wiki/pyOSC, Stand: 07.04.2016