Realisierung einer Noten-Tastatur

Um nun Töne kontrolliert erzeugen zu können braucht es eine Nutzeingabemöglichkeit das elektronische Musikinstrument spielen zu können. Für eine musikalische Ansteuerung bietet sich für Synthesizer eine Tastenreihe an.

Abbildung 1: Micromoog der Firma Moog

Wie in Abbildung 1 gesehen werden kann, besitzen auch analoge Synthesizer eine klavierähnliche Tastatur zur Musiknotenkontrolle. Diese Implementation eignet sich gut, da diese von sonstigen Tasteninstrumenten bekannt ist.

Die Wahl der Tasten

Elektronische Knöpfe eignen sich gut, um abzufragen, ob eine Note gespielt wird oder nicht. Doch wie diese Taste elektro-mechanisch realisiert wird, kann sehr unterschiedlich ausfallen und dies kann sich unteranderem auf Kosten, Nutzererfahrung und Platz auswirken.

Wir werden uns hier auf zwei Arten von Eingabemöglichkeiten befassen:

- **Mechanischer Taster als Schließer:** Der Klassiker unter den Tastern und wird auch am meisten angetroffen. Gerade im Bereich von DIY-Projekte sind sie beliebt aufgrund ihrer niedrigen Kosten und weiten Verfügbarkeit. Diese Art der Taster gibt es auch in vielen verschiedenen Variation in Bezug auf Haptik und weiteren Eigenschaften (Feedback, Geräusch, Drucktiefe, Bauform und -größe). Außerdem können sie grundsätzlich von jedem digitalen Eingangspin eines Microcontrollers erfasst werden.
- **CapSense / Touch-Pad:** Eine nicht so weit verbreitete Art von Kontakt ist die Möglichkeit einer Kapazitätsänderung. Dafür wird aber meist ein CapSense-fähiger Microcontroller benötigt oder ein externer IC-Baustein der CapSense erfassen kann. Die Realisation von diesen „Tastern“ wird durch freigelegte Leiterbahnen auf einem PCB realisiert die als Kontaktfläche dienen, daher sind diese Art der Taster sehr billig (nahezu null), wenn bereits ein PCB bei einem Hersteller angefertigt wird.

Da bereits ein Microcontroller gewählt worden ist, ein ESP32-S3, der die Möglichkeit hat Kapazitätsänderungen zu erfassen. Es sind auch digitale Eingangspins

vorhanden, um klassische Taster einzulesen. Für welche Variante man sich entscheidet steht noch offen, aber es wird in beiden Fällen mit einer Schaltungsmatrix gearbeitet, um die benötigten Drucksignale in ihrer Vollständigkeit einlesen zu können. Es wird mit einer Gesamtzahl von 24 Tasten (2 Oktaven) gearbeitet. Diese können in einer 6x4-Matrix (=24 Erkennungsmöglichkeiten) oder mit einer 5x5-Matrix (=25 Erkennungsmöglichkeiten) gearbeitet werden. In beiden Fällen wird dann statt 24 Pins nur noch 10 Pins (6+4 bzw. 5+5) am Microcontroller benötigt.

Probleme der Realisation mit der Kapazitätsmethode

Da nach näherer Betrachtung der Application Note zum Touch Sensor des ESP32 festgestellt werden kann, dass zwar oben genannte Sensoren in einem Matrixaufbau realisiert werden können, aber keine simultane Auswertung von zwei gleichzeitig gedrückten („berührten“) Sensoren ausgewertet werden kann. [1]

Daher eignet sich dieses System nicht für eine polyphone Ansteuerung eines Synthesizers, da Noteneingaben nur im monophonen Rahmen erkannt werden können. Weitere Punkte die gegen eine Realisation mit Touch Sensor als Keyboard sprechen sind unter anderem: Komplexes Hardwaredesign, komplexes Softwaredesign, kein haptisches Feedback, erhöhte Stör- und Fehleranfälligkeit.

Überlegungen und Realisation mit Tastern

Die konkrete Bauteilwahl der Taster kann im groben in ihrer Funktion und Bauweise grob bestimmt werden. Dabei sollen die gewählten Taster als Schließer dienen – sogenannte Kurzschlusstaster, da diese bei Tastendruck, die Verbindung kurzschließen. Daher kann grundsätzlich jeder generische Taster verwendet werden der als diese Eigenschaft besitzt. Bei der endgültigen Auswahl der Taster sollte dennoch auf eine wichtige Eigenschaft geachtet werden – die Haptik, da diese Eigenschaft die Benutzererfahrung am meisten beeinflusst in Bezug auf die Noteneingabe.

Wie bereits erwähnt wird die Tasteneingabe über eine Matrixschaltung in den Mikrokontroller eingelesen, um eine möglichst geringe Anzahl an GPIO-Pins zu verwenden. Dies ist wichtig, um noch weitere Bauelemente mit dem Mikrokontroller verknüpfen zu können.

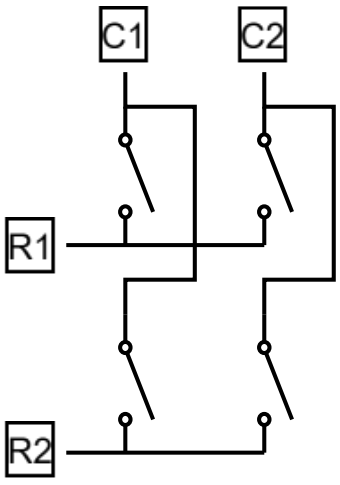

Abbildung 2: 2x2-Matrixschaltung

Die Pins einer Matrixschaltung kann man in 2 Arten unterscheiden: Scan-Pins und Activator-Pins. Dabei wird nachfolgenden Prinzip gearbeitet: Man legt fest ob die Reihen-Pins oder die Zeilen-Pins die Scanaufgabe übernehmen, wir bestimmen sie vorerst für die Zeilen-Pins (engl. rows). Da nun dadurch die Activatoraufgabe für die Spalten-Pins (engl. columns) bestimmt worden ist, kann mit dem Lesevorgang der Tasten begonnen werden. Es gibt gleich viele Lesevorgänge pro Zyklus, wie es Anzahl an Spalten gibt, in unserem konkreten Fall sind dies fünf Lesevorgänge pro Zyklus. Ein Zyklus sieht folgendermaßen aus: Alle Spalten-Pins werden hintereinander eingeschalten und wieder ausgeschalten, dabei sind alle Zeilen-Pins dauerhaft im Lesemodus (Input-Pulldown). Wenn nun ein Tastendruck erfolgt, kann dieser wie folgt ermittelt werden: Zeilen-Pin Nr. 1 erkennt Signal → Spalten-Pin Nr. 1 ist im Moment aktiv → Taste R1C1 ist somit gedrückt worden.

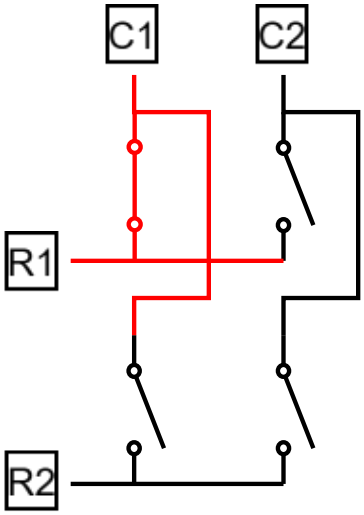

Abbildung 3: 2x2 Matrixschaltung einfacher Tastendruck

Somit können zwei Tastendrücke pro Zyklus problemlos ermittelt werden. Was passiert also, wenn mehr als zwei Tasten gedrückt werden?

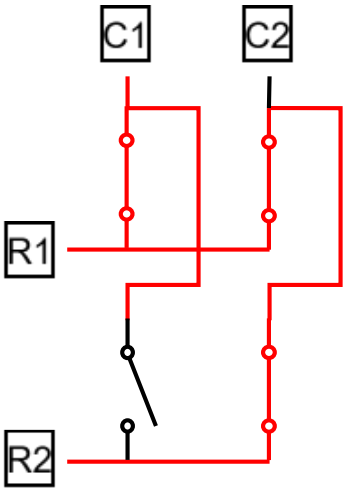

Abbildung 4: 2x2 Matrixschaltung dreifach Tastendruck

Als erwartenden Auswertung des Mikrokontrollers wird die Taste C1R1 erwartet. Da aber durch die geschlossenen Tasten in der Spalte C2 auch eine geschlossene

Verbindung zu R2 hergestellt worden ist, wertet der Mikrokontroller folgende Tasten aus: C1R1 und C1R2. Solch eine fehlerhaft Auslesung wird als Ghosting benannt. Die dabei fälschlicherweise ausgelesenen Taster werden als „Ghost-Keys“ (Geister Tasten) genannt. [2]

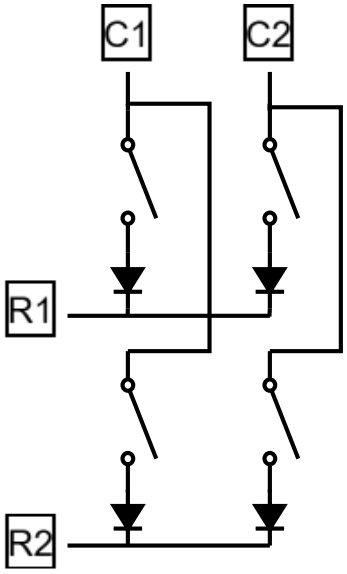

Abbildung 5: 2x2 Matrix mit Ghosting-Kompensation

Dieses Problem kann auf eine simple Art und Wiese behoben, in dem man die Stromflussrichtung in eine Richtung begrenzt. Ein Bauteil, das solche Eigenschaft aufweist, ist die allseits bekannte Diode. Wenn nun nach jedem Taster eine Diode geschalten wird, kann kein Ghosting mehr entstehen.[2], [3]

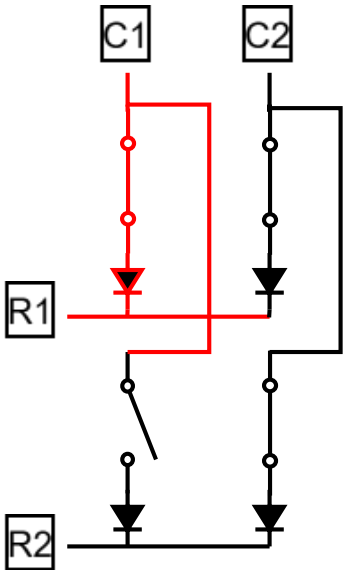

Abbildung 6: 2x2 Matrix beweise der Ghosting-Kompensation

Es kann beobachtet werden, dass wie erwartet nur die Taste C1R1 ausgelesen wird, obwohl drei Tasten gedrückt werden.

Interne Einheitenkommunikation

Hier wird nun die digitale Steuereinheit mit dem analogen Synthesizers-System verknüpft. Da der Synthesizer zur Ansteuerung der Noten einen bzw. zwei analoge Spannungspegel im Bereich 0V-5V verwendet, muss ein Digital-Analog-Wandler (DAW, auch englisch DAC) eingesetzt werden. Auf dem Markt gibt es dafür bereits kostengünstige IC-Bausteine, die die benötigt Auflösung besitzen. Es gibt Bausteine, die in einem IC mehrere DAC-Einheiten besitzen, daher sollte beachtet werden das mindestens zwei Einheiten vorhanden sind.

Überlegungen zur Bestimmung eines passenden IC

Es sollten einige Rahmenbedingungen gesetzt werden, um einen passenden Baustein zu finden.

Rahmenbedingung:

- Einfache Ansteuerungsmöglichkeit (z. Bsp. über ein Bussystem (I2C, …) oder serielle Datenübertragung)
- Passende Auflösung um analoge Werte in 1/12V Schritte von 0V bis 5V darstellen zu können. Eine Auflösung von 12-Bit sind bereits ausreichend, da dort bereits ein LSB zirka 1,22 mV (5/4096) entspricht. Ein 1/12V entspricht 83,3 mV.
- Mindestens zwei unabhängig ansteuerbare DAC-Einheiten für U_CV1 und U_CV2

Der gewählte IC-Baustein

Ein passender IC wurde nach kurzer Recherche bereits gefunden. Dabei wurde sich für den MCP4728 von Microchip entschieden. Dieser IC ist auch ein beliebter DAC bei Hobby-Elektronikern und DIY-Projekte. Ein kurzer Überblick zu den Fakten und Eigenschaften des DACs:

- Ansteuerung über I2C (Adressauswahl möglich)
- Vier getrennte Ausgangsspannungen (4 Kanäle)
- Auflösung von 12 Bit
- Interne oder externe Referenzspannung
- Spannungsversorgung von 2,7V bis 5,5V
- Arduino-Bibliothek von Adafruit zur Ansteuerung

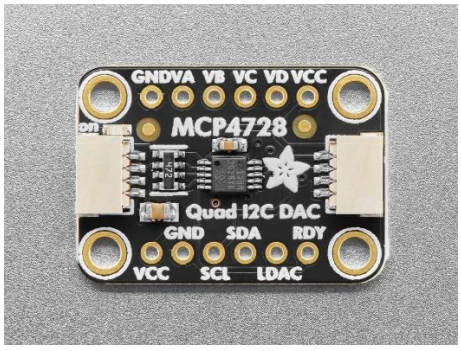

Abbildung 7: Entwicklermodul von Adafruit

Um erste Testversuche durchzuführen kann solch ein Modul gekauft werden, um im Steckbrettaufbau Schaltungskonzepte und -ideen überprüfen zu können.

Nähere Analyse zur Digital-Analog-Wandlung

Da hier in einer Wandlung nicht mit einer unendlichen Auflösung gearbeitet werden kann, entstehen in so einem System immer Abweichungen zwischen Ist-Wert und Soll-Wert. Daher eine kleine Analyse zur Wandlung:

$$U_{CV}(m) = \frac{m}{12}\ V$$

$$DAC(m) = \left\lfloor \frac{U_{CV}(m)}{U_{ref}} * (2^{12} - 1) \right\rfloor$$

$$U_{DAC}(m) = U_{ref} * \frac{DAC(m)}{2^{12} - 1}$$

$$U_{CV}, U_{DAC} \in \mathbb{R};\ DAC, m \in \mathbb{N}$$

Die Spannung U_CV ist der gewünschte Soll-Wert. Der Wert DAC ist die Wandlungszahl, die vom MCU an den DAC gegeben wird, dabei wird dieser exakt berechnet und anschließend auf die nächste Ganzzahl abgerundet.

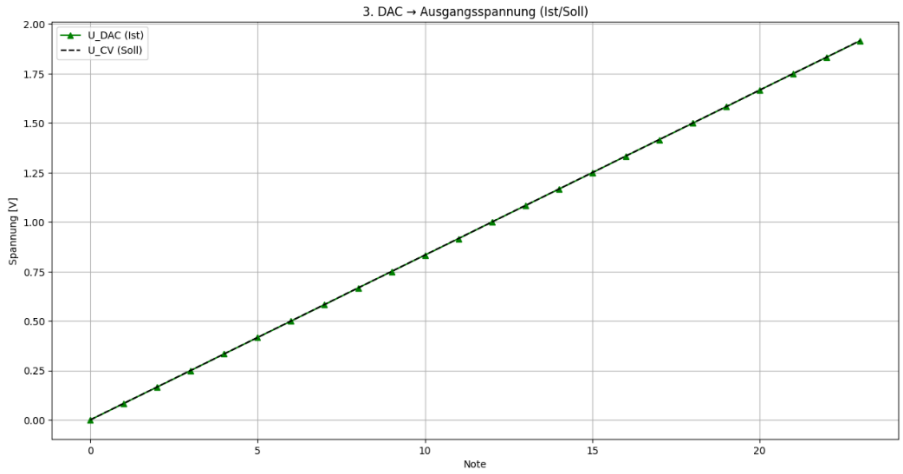

Abbildung 8: Vergleich der U_CV (Soll-Ist)

Die Soll-Ist Differenz zwischen den Größen ist nur minimal.

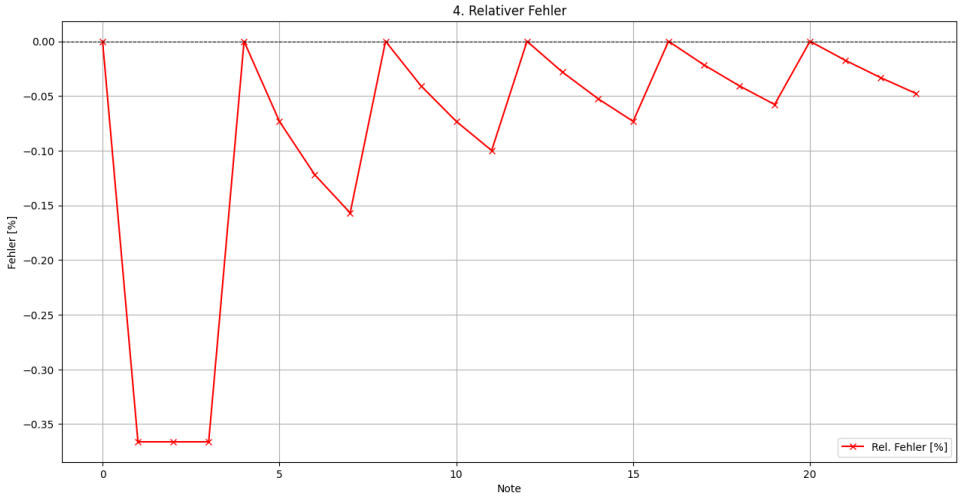

Abbildung 9: Relativer Fehler Soll-Ist

Wenn der Unterschied der Soll- und Ist-Werte als relativer Fehler betrachtet wird, ist ein Fehler von unter 5‰ erkennbar.

Abbildungsverzeichnis

Literaturverzeichnis

[1] „Touch Sensor Application Note“, Espressif Systems, Application Note. Zugegriffen: 10. August 2025. [Online]. Verfügbar unter: https://github.com/espressif/esp-iot-solution/blob/release/v1.0/documents/touch_pad_solution/touch_sensor_design_en.md

[2] Dave Dribin, „Keyboard Matrix Help“, Juni 2000. Zugegriffen: 10. August 2025. [Online]. Verfügbar unter: https://www.dribin.org/dave/keyboard/one_html/

[3] James Lewis, „Arduino Keyboard Matrix Code and Hardware Tutorial“, baldengineer. Zugegriffen: 10. August 2025. [Online]. Verfügbar unter: https://www.baldengineer.com/arduino-keyboard-matrix-tutorial.html