Das Thema meines Semesterprojektes war das Bauen einer barrierefreien Tastatur. Die Barrierefreiheit ist in diesem Fall durch eine Sprachausgabe bei Berührung der Tasten gegeben. Damit erfährt der Nutzer welche Taste er berührt, ohne diese auslösen zu müssen.

Um dies umzusetzen, habe ich die Funktion der Tastatur und die Funktion des Berührungssensors samt Audioausgabe in zwei Teile aufgeteilt. Passend dazu verwendet dieses Projekt zwei Microcontroller, einen Arduino Pro Micro und einen Raspberry Pi Pico. Der Pro Micro übernimmt dabei die Steuerung der Tastatur, während der Pico alles Weitere verarbeitet.

[redqueen Abbildung] Bei der Tastatur handelt es sich um die linke Hälfte eines Projektes zu ergonomischen Tastaturen, welches im Anhang verlinkt ist und hier abgebildet wird. [redqueen Plate stl Abbildung] Dem Projekt des Dritten habe ich die stl-Datei der Platte entnommen, welche ich mit einem 3D-Drucker gedruckt habe. Die Platte spielt eine wichtige Rolle beim Bau der Tastatur, da sie die gesamte Struktur der Tastatur bestimmt.

Ich hatte mich entschlossen, nur die linke Hälfte des Projektes nachzubauen, da die Tastatur damit vollkommen funktionsfähig ist und alle Buchstaben des ANSI US Layouts, für welches ich mich entschieden habe, abbilden kann. Diese Entscheidung habe ich getroffen, da das Bauen der zweiten Hälfte keine technische Komplexität hinzufügt und im Endeffekt redundant ist.

[Plate Switch Abbildung] In die Löcher der gedruckten Platte werden nun die mechanischen Schalter gedrückt. Ist die Platte gefüllt, kann man anfangen zu löten. Da die verwendeten Microcontroller über eine stark begrenzte Anzahl an nutzbaren Pins verfügt, ist es unmöglich jedem Schalter einen Pin zur Detektion des Auslösens zuzuweisen.

[Matrix Abbildung] Deshalb setzt man hier auf eine Matrix. Anstatt jedem Schalter einen Pin zuzuweisen, erhält somit jede Reihe und jede Spalte eine eigene Leitung. In der Praxis wird dann über jede Spalte Strom geschickt, während der Strom auf den Leitungen gemessen wird. Das heißt, dass die Spalten-Leitungen an als Output geschaltete Pins gelötet werden, während die Zeilen-Leitungen an als Input geschaltete Pins gelötet werden. Um nun zu bestimmen, welcher Spalte der Strom eines erfassten Signals entspringt, werden diese nacheinander an- und wieder ausgeschaltet.

[Baldengineer Dioden Abbildung] Des Weiteren muss darauf geachtet werden, dass jeder Schalter-Pin über den Strom empfangen wird mit einer Diode versehen wird. Die Dioden dienen der Prävention eines Phänomens namens „ghosting“. Wenn Ghosting geschieht, können nicht vorgesehene Knopfdrücke innerhalb der Matrix entstehen. Der Grund dafür ist, dass der Strom beim Drücken mehrerer Tasten auf alle Leitungen fließen kann, was es unmöglich macht den genauen Ursprung zu bestimmen. Damit der Strom auf den vorgesehenen Leitungen bleibt, verwendet man Dioden.

[Fertig gelötet Abbildung] In diesem Beispiel habe ich die Dioden an den linken Pin der Schalter und die Stromversorgung über Kupferlackdraht an den rechten Pin gelötet. Da die Drähte der Dioden lang genug sind, kann man die einzelnen Dioden auf einer Reihe miteinander verbinden, um die Leitung für die Reihe der Tastatur zu erhalten. Der Kupferlackdraht wird an jeden rechten Pin der Schalter in einer Spalte gelötet. Ist jeder Pin mit einer Spalten- und einer Zeilen-Leitung verbunden, können die Leitungen an die Pins des Arduinos gelötet werden.

[Papierisolierung Abbildung] Zur recht simplen Isolierung von Arduino Pins und Schaltern, habe ich ein A6 Blatt Papier zwischen den Schaltern der Tastatur und Arduino gelegt.

[Pro Micro Pinout Abbildung] Bei den Pins sollte man lediglich darauf achten, dass die Leitungen nur an GPIO Pins gelötet werden. Empfehlenswert ist es, die Leitungen der Spalten und Zeilen auf beide Seiten der Arduino Pins aufzuteilen.

[Layout Editor Abbildung] Ist man mit dem Löten fertig, muss man nun die Firmware erstellen. Da ich das Grundkonzept einer Tastatur später auf dem Raspberry Pi Pico umsetze und ich weitere Funktionen benötige, die in der Arduino IDE umständlich in der Realisierung sind, habe ich die Firmware mittels verschiedener Services generieren lassen.

Der erste Schritt zum Erstellen der Firmware ist es das Layout seiner Tastatur in einem Layouteditor zu erzeugen. Ist das generelle Layout vervollständigt, kann man noch die gewünschten Tastenbelegungen festlegen. Ist man zufrieden, muss man das Layout in Form einer .JSON-Datei herunterladen.

[Firmware Builder Abbildung] Die heruntergeladene .JSON-Datei muss nun in ein Tool zur Generierung der Firmware eingespielt werden. Die Pfade, die die Leitungen der Zeilen und Spalten annehmen sollen, können hierbei überarbeitet werden, wenn sie mit der gelöteten Hardware nicht übereinstimmen. Stimmt die Tastaturenmatrix auf der Website mit der Matrix der Hardware überein, kann der nächste Schritt eingeleitet werden. Unter dem Tab „Pins“ kriegen die Leitungen ihren zugehörigen Pin zugewiesen. Wichtig ist hierbei, dass die Pins auf der Website nicht an die GPIO Pins des Arduino Pro Micros angepasst sind, sondern an den jeweiligen Controllerchip des Microcontrollers. Da der Arduino Pro Micro einen ATMega32U4 besitzt, wählt man diesen aus und konvertiert die GPIO Pins zum AVR Format. [ProMicro Pinout Abbildung] Dafür gibt es diese Pinout Grafik.

Da ich nur eine Hälfte der Tastatur gebaut habe, brauche ich mindestens eine weitere Ebene, welche meine Tastendrücke übersetzt. Unter „Keymap“ sieht man das Layout der eigenen Tastatur. Die einzelnen abgebildeten Tasten können hier, wenn noch nicht vorher geschehen, konfiguriert werden.  
Der unten linken Taste meiner Tastatur weise ich die Funktion MO(1) zu. MO(1) steht für „MOmentarity turn on layer, when pressed“, wobei durch die Nummer eins in der Klammer zur Ebene 1 während des Tastendrucks gewechselt wird. Die Standardebene ist die Ebene 0.   
Die Tasten der Ebene 1 konfiguriere ich so, dass ich alle wesentlichen Funktionen einer Tastatur und alle Buchstaben im ANSI US Layout verwenden kann.

Letztlich bleibt nur noch das Kompilieren der Firmware unter „Compile“. Hier sollte man die .zip Datei herunterladen, damit man die Firmware vor dem Flashen auf den Arduino überarbeiten kann.

Das Überarbeiten besteht lediglich daraus, Teile des generierten Codes in der Keymapdatei der zip zu löschen, um Speicherplatz zu sparen.

Mittels eines Tools namens QMK kann man die Keymap nun kompilieren, worauf hin man eine .hex Datei erhält. Diese .hex Datei ist unsere Firmware und kann nun auf den ATMega32U4 geflashed werden.

Dazu verwendet man die sogenannte QMK Toolbox. Hier muss man nun die Firmware und den Microcontroller auswählen. Beim Anschließen des Pro Micros sollte dieser nun erkannt werden und die Möglichkeit zum flashen sollte verfügbar sein.

War das Flashen erfolgreich, sollte die Hardware der Tastatur nun auch als solche funktionieren.

Zum Testen der Tastaturfunktionen verwende ich den Key Tester der Software VIA.

Zur Erkennung von Berührungen habe ich zunächst eine Matrix aus kapazitiven Sensoren aufgebaut. Da ich an dem Pro Micro nicht genügend Pins hatte, habe ich einen Raspberry Pi Pico und die Sensoren samt Audioausgang separat von der Tastatur zusammengestellt.

Der kapazitive Sensor funktioniert, indem ein Stück Metall, welches unter Strom steht, mit dem Finger des Anwenders einen Kondensator bildet. Je näher der Finger an den Sensor kommt, desto stärker steigt die Kapazität des Kondensators an. In diesem Fall habe ich Kupferlackdrähte als Sensoren verwendet, welche über 5,1 Megaohm Widerstände an den Output Pin des Pico angeschlossen werden. Da ich mit der Sensorik eine Tastatur emulieren möchte, habe ich auch hier wieder zu einer Matrix gegriffen. Der Unterschied zu der Arduinotastatur besteht darin, dass es hier eine Stromquelle gibt, die von allen Leitungen geteilt wird. Auch hier sind die Leitungen wieder in Zeilen und Spalten aufgeteilt. Jede der Leitungen ist über einen Input Pin am Pico angeschlossen.

Zu jedem Zeitpunkt fließt auf allen Leitungen Strom, damit wir eine Veränderung in der Kapazität des Kondensators erkennen können. Die gemessenen Werte werden in einer Matrix aufgenommen, welche alle Kombinationen der Werte der Spalten und Zeilen zusammenaddiert.