1. 7-Segment-Anzeige programmieren

Wir besprechen einen Sketch, der eine Anzeige im Sekundentakt hochzählen lässt. Hier dient der Arduino als Decoder und Treiber für die LED-Segmente. Zusätzlich beinhaltet er einen Sekundenzähler.

a) Ändere den Sketch so ab, dass die Anzeige rückwärts zählt.

b) Ändere den Sketch noch mal ab, sodass die Anzeige im Sekundentakt hochzählt und dann 4-mal so schnell runter zählt. Zusätzlich soll bei jedem Hochzählen der Anzeige ein ultrakurzer tiefer Ton und beim Runterzählen ein entsprechend höherer Ton zu hören sein.

c) Erweitere die vorwärtszählende Anzeige so, dass nach "9" folgende Zeichen angezeigt werden: A, b, C, d , E, F dann soll es wieder bei "0" beginnen.

d) Verändere den Sketch so, dass das Zählen nur auf Tastendruck hin geschieht.

2. Der BCD-Decoder (**B**inary **C**oded **D**ecimals)

Für das obige Beispiel benötigen wir 7 Ausgangpins des Arduino. Mithilfe eines Decoders reichen 4 Pins aus.
Du benötigst also das Decoder-IC **CA3161E** und die große 7-Segment-Anzeige.
Verdrahte den Sketch (nachdem der Arduino gelöscht und dann das USB-Kabel entfernt wurden.)
Ergänze den Sketch BCD Decoder auf dem USB-Stick sodass alle 10 Ziffern nacheinander angezeigt werden. Bitte die Änderungen des Sketches **nicht abspeichern!**

a) Die Anzeige soll auf Tastendruck eins weiterzählen. Verändere nun das Delay von 300ms auf 30ms und beobachte die Auswirkung.

b) Ändere den Sketch so ab, dass die Anzeige im Sekundentakt „automatisch" läuft.

c) Lass unsere Binärzahl nicht bis 9 sondern bis 15 zählen. Was wird dann angezeigt? Haben wir darauf einen Einfluss? Vergleiche mit Aufgabe 1c)

d) Verwende unser **Sprachmodul ISD1820** und lasse damit eine der Ziffern zusätzlich per Text ansagen.

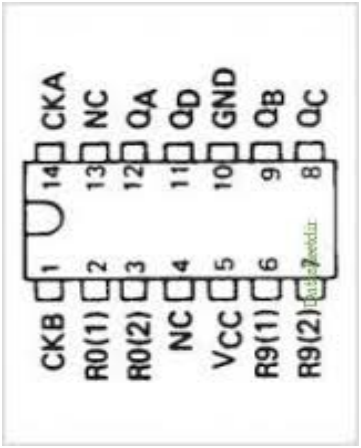
Erweiterung: Stelle die Ausgänge des Arduino an den BCD-Decoder mit dem Seriellen-Monitor dar und zwar in der Reihenfolge DCBA. Beachte: „A" ist das niederwertigste Bit. Dann siehst du auf dem Monitor die Binärzahl und auf der Anzeige die dazugehörige Dezimalzahl.

Hier die wesentlichen Schritte des Sketches:

- Den Variablen A bis D (=Eingänge des Decoders) die Integer-Zahlen 4 bis 7 zuordnen, der Taste die 8 (=Pinbelegungen).
- Im Setup den pinMode (INPUT/OUTPUT) festlegen
- Nach dem Setup die Zählervariable festlegen, die mit Null startet. Im Loop abfragen, ob der Taster gedrückt (=LOW) ist, falls ja muss die Zählervariable um eins erhöht werden. In dieser if-Abfrage noch eine if-Abfrage einbauen, die den Zähler zurücksetzt falls er auf 10 steht.
- Nun kommen die if-Abfragen für alle Ziffern. Wenn der Zähler z.B. bei „3" steht (=0011) müssen A und B durch digitalWrite auf HIGH und C, D auf LOW gesetzt werden.

3. Hardware-Zähler (optional)

Jetzt tauchen wir kurz in die reine Hardware ein und realisieren den Zähler mit Hilfe des Zähler-ICs **SN7490**. Zur Verkabelung benötigst du die volle Konzentration! V_{CC} entspricht +5V, CK_B muss mit Q_A verbunden werden, CK_A ist der Takteingang (Clock) und Q_A bis Q_D sind die Ausgänge, die mit den Eingängen des Decoders verbunden werden (Q_A = 2^0 usw.). Jeweils einer der Rückstelleingänge R_0 bzw. R_9 muss mit GND verbunden werden.



Baue einen solchen Zähler auf und teste ihn. Du wirst sehen, dass der Zähler Ziffern überspringt. Das liegt am unvermeidlichen „Prellen" des Tasters. Nun hilft uns wieder der Arduino mit dem wir den Taster per Sketch „entprellen" können. Ein solches „Micro-Programm" solltest du inzwischen selbst schreiben können ☺ (Tipp: Beim „BCD-Decoder"-Sketch kannst du notfalls spicken).

Erweiterung: Mit Hilfe der Rückstelleingänge kannst du den Zähler so abändern, dass er z.B. nur bis 6 oder 8 zählt und dann wieder von vorne beginnt. Dazu ist kein Sketch notwendig, schließlich arbeiten wir hier ausnahmsweise im reinen Hardwarebereich.