1. Mit Taster (Button) schalten.....

a) Lade den Sketch „Button" (Beispiele-02Digital) auf den Arduino. Trenne dann das USB Kabel vom Arduino, installierte die LED an Pin13 und den Button (Taster) an Pin2. Verwende den Shield-Button der LOW wird, wenn man den Taster drückt. Daher muss der INPUT noch mit PULLUP ergänzt werden. Die LED soll bei gedrücktem Button leuchten.
b) Prüfe anschließend, ob folgender Button-mini-Sketch auch funktioniert. Vergleiche mit dem original-Sketch. Hier wird wieder der Shield-Button verwendet.

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(2, INPUT_PULLUP);
}
void loop() {
  if (digitalRead(2)==LOW) {
    digitalWrite(13, HIGH);
  } else {
    digitalWrite(13, LOW);
  }
}
```

c) Verwende nun den **Joystick-Taster** und erweitere den Sketch so, dass beim Drücken des Tasters eine sehr kurze akustische Rückmeldung aus drei aufeinander-folgenden Tönen erzeugt wird und die LED nach einer Zeitverzögerung von ca. 4 Sekunden ausgeht.

d) Für Profis: Versuche nun den Sketch nochmals zu erweitern, so dass die LED erst nach dreimal Drücken des Tasters für 4 Sekunden leuchtet.

2. Elektronisches Keyboard

Zunächst müssen wir die Library **pitches.h** etwas verstehen. Damit lässt sich eine Klaviatur simulieren. Diese Library (Bibliothek) wird hier inkludiert.

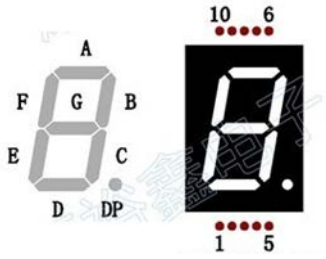
Wir wollen mit z.B. drei Tastern verschiedene Töne abspielen. Folgende Probleme sind dabei zu lösen:
Es müssen dauernd die Taster abgefragt werden, ob ihr Zustand sich ändert. Ist das bei einem Taster der Fall, so muss ein bestimmter Ton aus einem Array abgespielt werden. Öffne dazu den Sketch aus den Beispielen (02digital) **toneKeyboard**. Dieser ist für Touch-Sensoren aufgebaut, die wir (noch) nicht haben. Wir verwenden Buttons, die an digitale Eingänge 0,1,2 (besser wäre 2,3,4) angeschlossen und mit **INPUT_PULLUP** auf einen HIGH-Level gezogen werden, falls der Button nicht gedrückt ist. Ansonsten hätte der Digital-Eingang einen nicht definierten Zustand.
Verändere also den Arduino-Sketch und teste ihn.

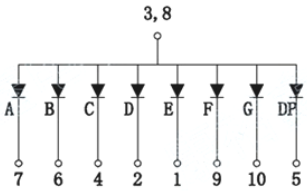
Ausblick: Später verwenden wir das Modul LED&KEY und können mit 8 Tastern zumindest eine Oktave abdecken. Wir kommen also auf das Keyboard zurück.

3. 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.

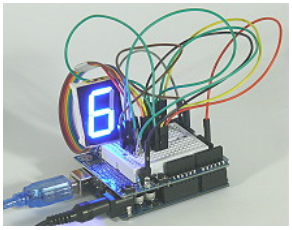
a) Ändere den Sketch so ab, dass die Anzeige rückwärts zählt.
b) Ändere den Sketch noch mal ab, so dass die Anzeige im Sekundentakt hochzählt und dann 4-mal so schnell runter zählt. Zusätzlich soll bei jedem Hochzählen der Anzeige ein ultrakurzer tiefer Ton und beim Runterzählen ein entsprechend hoher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



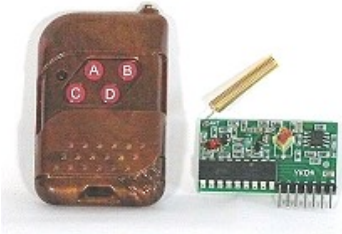


c) Erweitere die vorwärts-zählende Anzeige so, dass nach "9" folgende Zeichen angezeigt werden:
A, b, C, d , E, F dann soll es wieder bei "0" beginnen.

d) Setze an geeigneter Stelle ein delay(), so dass du die Entstehung jeder Ziffer (oder jedes Zeichens) quasi in Zeitlupe beobachten kannst. Entferne dann das delay() wieder.



e) Für Profis: Verändere den Sketch so, dass das Zählen nur auf Tastendruck hin geschieht.



f) Verwende die Funk-Fernbedienung.
Mit Kanal **A** soll hochgezählt werden, Kanal **B** soll die Anzeige auf 0 zurücksetzen.
Wer sich´s zutraut:
Mit Kanal **C** soll rückwärts gezählt werden.