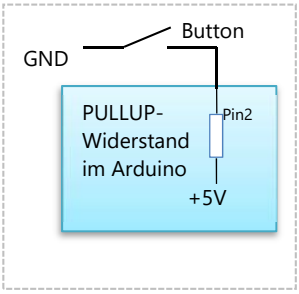
1. Mit Taster (Button) schalten.....

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.

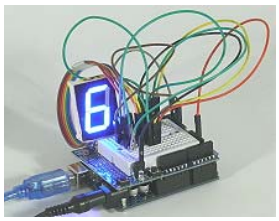
2. Elektronisches Keyboard (I)

Zunächst müssen wir die Library **`pitches.h`** etwas verstehen. Damit lässt sich eine Klaviatur simulieren. Diese Library (Bibliothek) wird hier inkludiert. Sie ordnet jeder Taste auf dem Keyboard die zugehörige Frequenz zu. Du gibst nur noch die Noten ein und die richtige Tonfrequenz wird ausgewählt. Wir wollen mit drei Tastern verschiedene Töne abspielen. Dabei gibt es zwei Versionen:
a) Nach dem Drücken einer Taste wird der Ton für z.B. 0,2 Sekunden gespielt.
b) Der Ton läuft so lange, wie die Taste gedrückt bleibt.

Löse wenigstens eine der beiden Aufgaben. a) lässt sich simpel durch if-Abfragen lösen, b) mit Hilfe von Schleifen und Arrays und natürlich einer if-Abfrage. Hierfür gibt es ein Extrablatt (Analysen).

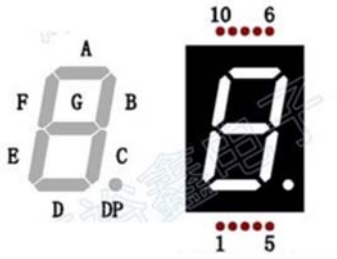
Im Loop ist der Zustand der drei Buttons abzufragen. Wir verwenden die digitale Eingänge 2,3,4 die mit **`INPUT_PULLUP`** auf einen HIGH-Level gezogen werden, falls der Button nicht gedrückt ist.

3. 7-Segment-Anzeige programmieren

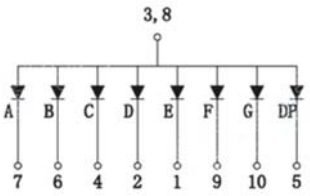


Wir besprechen einen Sketch, der eine Anzeige im Sekundentakt hochzählen lässt. Hier dient der Arduino als Decoder und Treiber für die LED-Segmente.

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 ultra-kurzer 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 das delay() anschließend wieder.

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

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.

Info: **busy-waiting** (aktives Warten). Leere Schleifen, also solche, bei welchen nichts im Anweisungsblock steht (den man dann auch weglassen kann) bewirken eine „Blockade" des Programms. **`while(digitalRead(button)==HIGH);`** bedeutet: Tue nichts, solange die Bedingung „button HIGH" erfüllt ist. Wird die Aussage digitalRead(button)==HIGH falsch, so läuft das Programm weiter. Das kann man auch zur Lösung der Aufgabe 3e) verwenden.