1. Vom Ton zur Melodie

Öffne in 02Digital den Sketch **toneMelody**.
Verkabele die Hardware und lasse den Sketch laufen.
Gestalte nun deine eigene Melodie.

2. Gewürfelte Töne

Es sollen sehr kurze Töne nach dem Zufallsprinzip **random(x,y);** abgespielt werden. Der Frequenzumfang beläuft sich von z.B. 300Hz bis 1500Hz.

Erzeuge einen Sketch und teste ihn (bitte nur kurz!! ☺).
Beachte: Mit **tone(x,y);** läuft ein Ton an Pin(x) mit der Frequenz(y) so lange, bis die Anweisung **noTone(x);** oder eine neue Tonanweisung kommt.

3. Akustische Sirene

Programmiere einen Arduino so, dass er zunächst ein anschwellender Sirenenton zu hören ist.
Erweitere den Sketch nun geeignet um einen „richtigen" Sirenenton zu erzeugen (300 bis 800Hz).....nicht nerven! ☺

4. Bei Dämmerung einschalten!

Jede autonom gesteuerte Außenbeleuchtung hat einen

LDR (**L**ight **D**ependend **R**esistor) und einen Wärmesensor um die Lichtquelle zu schalten. Wir wollen zunächst einen Sketch entwickeln, mit welchem eine LED dann eingeschaltet wird, wenn die Umgebungshelligkeit einen bestimmten einstellbaren Wert unterschreitet. Dann soll sie auch mindestens 5 Sekunden lang an bleiben. Dazu legen wir an den analogen Eingang A0 einen LDR gegen +5V und einen Widerstand von 47KOhm (gelb-violett-orange) gegen GND. Am analogen Eingang A1 schließen wir über ein Poti eine (Referenz-)Spannung an, die man zwischen 0V und 5V einstellen kann.

Definiere nun: int LED=12; int LDR=A0; int Poti=A1;
Was gehört ins Setup? Bedenke: Die Pin-Modes an analogen Eingängen müssen nicht deklariert werden, das können ja nur Eingänge sein ☺ .
Im Loop musst du nun über eine if-Anweisung die analogRead-Werte vom LDR und vom Poti vergleichen. Wenn der Wert an A0 kleiner als an A1 ist, wird die LED auf HIGH geschaltet.

5. Töne und noch mehr Töne

Öffne den Beispiel-Sketch **tonePitchFollower** (Tonhöhenfolger) in 02Digital. Baue die Schaltung laut Anweisung auf. Der 4,7K-Ohm Widerstand hat die Farbcodierung *gelb-violett-rot*. Ersetze zunächst in der Anweisung **tone(9, thisPitch, 10);** **thisPitch** durch **sensorReading**. Lasse den Sketch laufen und verdunkele den Fotowiderstand und beobachte dabei auch den seriellen Monitor.
Verwende nun den originalen Sketch und überlege, was die **map**-Anweisung bewirkt

6. Einführung des Shift-Registers

Die seriell-parallel-Umsetzung ist eine ganz zentrale Aktion bei der Programmierung. Nur so ist es möglich z.B. mit nur 3 Datenpins des Arduino beliebig viele Ziffern in ein Anzeigenmodul zu transportieren.
Öffne dazu das separate Dokument „74HC595 8 Bit Schieberegister mit Latch".

a) Der **Sketch 1** befindet sich auf dem Stick. Teste ihn und nimm Veränderungen dran vor.

b) Erzeuge damit ein schnelles „Lauflicht". Hier sollten nur wenige zusammenhängende Bits aktiv sein, z.B. 10000000.

c) Vereinfache den Sketch, so dass ein im Array vorgegebenes Bitmuster nach dem Hochladen des Sketches sofort von den LEDs angezeigt wird. Dazu muss man das Register nicht löschen und die Übergabe an das Latch muss nur noch erfolgen nachdem das Shift-Register geladen wurde, also nach der for-Schleife.

d) Für Profis: Erweitere den Sketch so, dass die Geschwindigkeit des Lauflichtes langsam zu- oder abnimmt.

e) Nur für Profis! Verändere den Sketch noch einmal, so dass das Lauflicht reflektiert wird, also hin und her läuft. Mit den Möglichkeiten aus Sketch 1 ist das recht schwierig! In Aufg. 1 aus Blatt 6 werden wir hierfür eine einfache Methode kennenlernen.

7. Spielen mit dem Shift-Register

.....o.k., dabei lernt man auch viel!
Lade den **Sketch 2** und Teste ihn.

a) Verändere ihn so, dass rückwärts gezählt wird.

b) Erweitere den Sketch so, dass im seriellen Monitor zusätzlich der Wert der Variablen „counter" erst als Binärzahl und ca. 5-10 Leerzeichen dahinter als Dezimalzahl angegeben wird. New Line nicht vergessen!

Binäre Anzeige:	Serial.print(counter, BIN),
Dezimale Anzeige:	Serial.print(counter)