

Hier folgen weitere kleine Sketche um die Anwendung der Arduino-Software („C“) zu üben:

1. Mit „Fade“ dimmen, Mischfarben erzeugen

a) Öffne den Sketch „fade“ auf dem Stick, teste ihn und versuche diesen zu verstehen. Verändere die Parameter und vergleiche. Mache dir nochmal klar, was PWM bedeutet. Schließe dazu parallel zur LED das Digital-Oszilloskop an und beobachte das Display. Das schwarze Kabel muss dabei an GND angeschlossen werden.



b) Erweitere den fade-Sketch für die 3-Farben-LED. Dabei sollte jede Farbe extra in der Intensität hoch- und runtergefahren werden. „Bunt“ wird das Licht nur, wenn die 3 LEDs nicht synchron laufen. Dies

gelingt am besten, wenn fadeAmount für alle 3 Farben im Bereich von 3 bis 6 mit random(x,y); zufällig gewählt werden.

x: niedrigster int-Wert, y: höchster int-Wert. Damit erhält man alle denkbaren Mischfarben.

Tipp: Es gibt jetzt nicht nur die Variable „brightness“, sondern „brightnessRot“, „brightnessGruen“ usw. Anstelle von fadeAmount=5; schreibst du fadeAmountRot = random(3,6); usw. Der Änderungsbetrag wird also zufällig gewählt. Im Prinzip verdreifacht sich der Sketchaufwand, was aber durch Kopieren nicht so schlimm ist.

2. Vom Ton zur Melodie

Öffne den Sketch toneMelody auf dem Stick und lass ihn laufen. Gestalte nun deine eigene Melodie.

3. Gewürfelte Töne

Es sollen sehr kurze Töne nach dem Zufallsprinzip random(x,y); abgespielt werden. Der Frequenzumfang beläuft sich von z.B. 100Hz bis 2000Hz. Erzeuge einen Sketch und teste ihn (bitte nur kurz!! ☺).

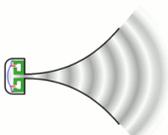


1. Zusatz: Das Würfeln soll langsam beginnen (200ms) und immer schneller werden. Empfohlene

Änderungsrate: 0,4ms. Irgendwann sind die Töne nicht mehr wahrnehmbar und es beginnt das „Rosa-Rauschen“.

2. Zusatz: Erweitere den Sketch so, dass eine vorher einstellbare Zahl von Tönen nicht überschritten wird.

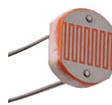
4. Akustische Sirene



Programmiere einen Arduino so, dass zunächst ein anschwellender Sirenenton zu hören ist. Erweitere den Sketch nun geeignet um einen „richtigen“ Sirenenton zu erzeugen

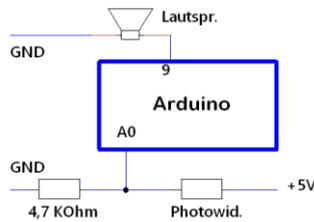
(300 bis 800Hz).....nicht nerven! ☺ Bei „Fade“ (Aufg. 1) kannst du spicken!

5. Töne und noch mehr Töne



Hier arbeiten wir mit einem LDR-Widerstand (Light Dependend Resistor), dessen Wert umso kleiner wird, je mehr Licht auf ihn trifft.

Öffne den Beispiel-Sketch tonePitchFollower (Tonhöhenfolger) auf dem Stick. Baue die Schaltung 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, verdunkle den Fotowiderstand und beobachte dabei auch den seriellen Monitor und Plotter.

Verwende nun den originalen Sketch und überlege, was die map-Funktion bewirkt.

Info: Was ist eine map-Funktion?

Mit Hilfe dieser Funktion können wir z.B. Sensorwerte so umrechnen, dass sie für unseren Zweck passen. Wenn z.B. ein Temperaturfühler Werte zwischen 0 und 1023 ausgibt und wir wissen, dass die Grenze 20 und 100 ist, kann map das für uns übernehmen. Syntax:

```
y = map(x, 0, 1023, 20, 100);
```

x ist die Variable, die transformiert werden soll.

6. Lichtgesteuerte Ampel

Nur ein paar kleine Änderungen und Zusätze und schon wird aus dem tonePitchFollower eine helligkeitsgesteuerte Verkehrsampel. Man kann auch beides kombinieren. Dann wäre die Ampel sogar blindengerecht.



Der Sketch soll folgendes bewirken:

Bei Dunkelheit soll die Ampel rot leuchten, bei Dämmerung gelb und wenn es hell ist grün. Dabei soll leise ein tiefer, mittlerer, bzw. hoher Ton zu hören sein. Verwende 3

separate LEDs. Widerstände nicht vergessen!

Installiere auch hier den Seriellen Monitor für den analogen Eingang A0. Durch Beobachten der Werte beim Verdunkeln des LDR kannst du die Übergänge (rot/gelb/grün) sinnvoll festlegen.