Übung 10: Zustandsautomaten

Hardware:

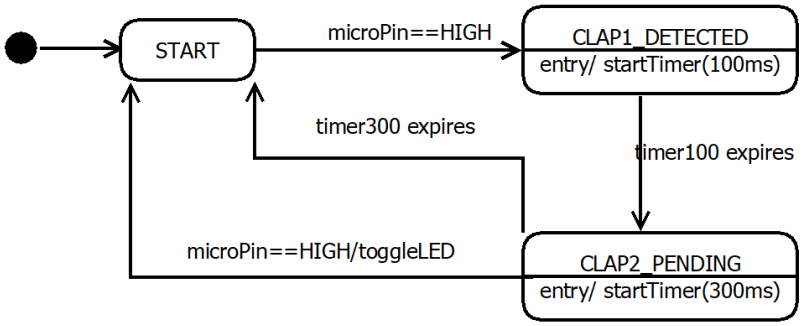
- Basis: Arduino, Steckbrett, Kabel
- IDUINO SE019 Mikrofonsensor
- Blaue LED + 100 Ohm Vorwiderstand

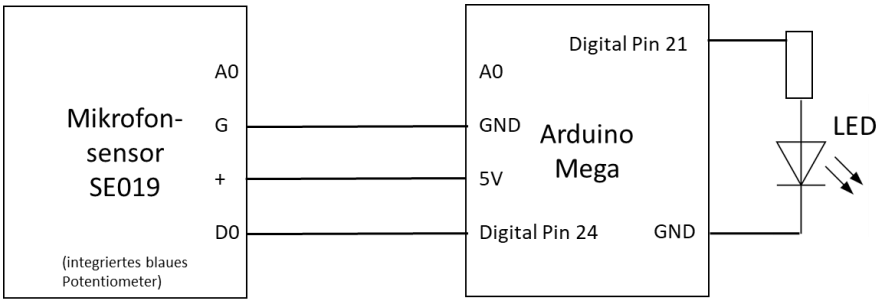
Mikrofonsensor 

Informationen:

- Datenblatt IDUINO (mitgeliefert) und Datenblatt ATmega2560:
- Pin Mapping: https://www.arduino.cc/en/Hacking/PinMapping2560

Sie implementieren einen **Zustandsautomaten** zum Erkennen eines „doppelten Klatschens“. Man könnte so z.B. eine Fernsteuerung für einen Lichtschalter realisieren. Nach dem 1. Klatschen müssen Sie mindestens 100 ms warten, um das 1. Klatschen sauber vom 2. Klatschen zu trennen. Spätestens 300 ms nach dem 1. Klatschen muss jedoch das 2. Klatschen erfolgen, ansonsten ist es kein Doppelklatschen.





Aufgabe 1: Schaltung, Kalibrierung des Mikrofons

Das Mikrofon ***SE019*** liefert an Pin `D0` die Ausgabe `HIGH`, falls der Schallpegel über einem Schwellwert liegt. Der Schwellwert lässt sich mit einem Schraubendreher am blauen Potentiometer einstellen[1].

a) Bauen Sie die die abgebildete Schaltung auf!

b) Der folgende Code ist als Beispielprojekt `01_SensorTuning` vorgegeben. Verwenden Sie den Code für die Kalibrierung, also die Einstellung des Schwellwertes. **Vorgehen unbedingt beachten!!!**
   - Drehen Sie ggfs. erst im Uhrzeigersinn bis die blaue LED leuchtet.
   - Drehen Sie dann gegen den Uhrzeigersinn bis die grüne LED ***gerade so*** erlischt.

[1] Seien Sie erfinderisch, falls Sie keinen kleinen Schraubenzieher haben.

Code zur Einstellung der Sensitivität des Mikrofonsensors

```
int microPin = 24;
int ledPin = 21;

void setup() {
  pinMode(microPin, INPUT);
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, LOW);
}

void loop() {
  int val = digitalRead(microPin);
  if (val == HIGH) {
     digitalWrite (ledPin, HIGH);
  } else {
     digitalWrite (ledPin, LOW);
  }
}
```

Aufgabe 2: Implementierung über Case-Struktur

a) Setzen Sie den Zustandsautomaten mit Hilfe einer ***switch-case*** Anweisung um. Testen Sie!
- Verwenden Sie das vorgegebene Beispielprojekt `02_CaseStruktur.`
- Die Zustandsnamen sind als `enum` deklariert und entsprechen der Abbildung!
- Für jeden Zustand (`case`-Anweisung) müssen Sie auf relevante Ereignisse testen, siehe Vorlesung. Orientieren Sie sich an der Abbildung des Zustandsautomaten.
- Timer-Ereignisse dürfen ***nicht*** mit einem `delay(.)` implementiert werden. Gehen Sie vor wie in der Vorlesung und halten Sie für jedes Timerereignis eine ***eigene (!)*** Variable, die die Startzeit des Timers speichert → `millis()`. Keine Interrupts!
- Sie dürfen beliebige Arduino-Kommandos verwenden.

b) Wie groß ist das hochgeladene Programm in Byte? Ggfs. erneut kompilieren.

Aufgabe 3: Implementierung über Tabelle

a) Implementieren Sie den Zustandsautomaten mit einer ***Tabelle von Funktionszeigern*** und testen Sie!
- Verwenden Sie das vorgegebene Beispielprojekt `03_Tabelle.`
- Zusätzlich zu den Zuständen sind nun auch die Namen der Ereignisse als `enum` vorgegeben und deklariert.
- Tragen Sie in das vorgegebene 2-dimensionale Array (= Tabelle), die korrekten Funktionen ein. Um die Arbeit zu erleichtern, sind nötige Funktionen weiter oben im Code bereits implementiert.
- Implementieren Sie die Methode `state_machine().` Dort muss zu Beginn nach aufgetretenen Ereignissen getestet werden, dann der Zustand weitergeschaltet werden. Orientieren Sie sich an der Vorlesung.
- Timer-Ereignisse dürfen weiterhin ***nicht*** mit einem `delay(.)` implementiert werden. Gehen Sie vor wie in Aufgabe 2).
- Sie dürfen beliebige Arduino-Kommandos verwenden.

b) Wie groß ist das hochgeladene Programm? Vergleichen Sie mit Aufgabe 2).

c) Wann würden Sie eine Implementierung mit Funktionszeigern vorziehen? Wann eine Implementierung mit `switch-case`?