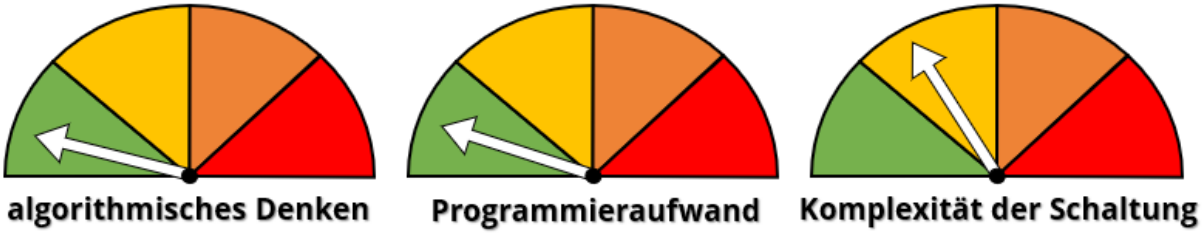
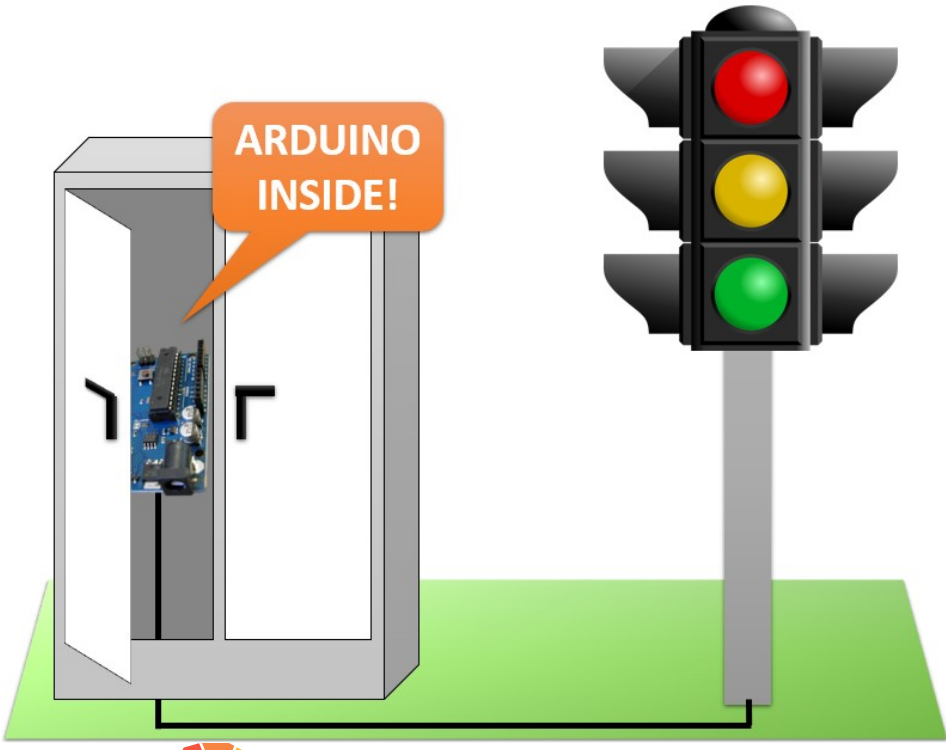
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Station 5 | Halt Stop! – Jetzt fahr ich!





Professur für Didaktik der Informatik

SYLBER Synergetische Lehrerbildung im exzellenten Rahmen

Das Maßnahmenpaket „TUD-Sylber² – Synergetische Lehrerbildung im exzellenten Rahmen“ wird im Rahmen der gemeinsamen „Qualitätsoffensive Lehrerbildung“ von Bund und Ländern aus Mitteln des Bundesministeriums für Bildung und Forschung gefördert.

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung

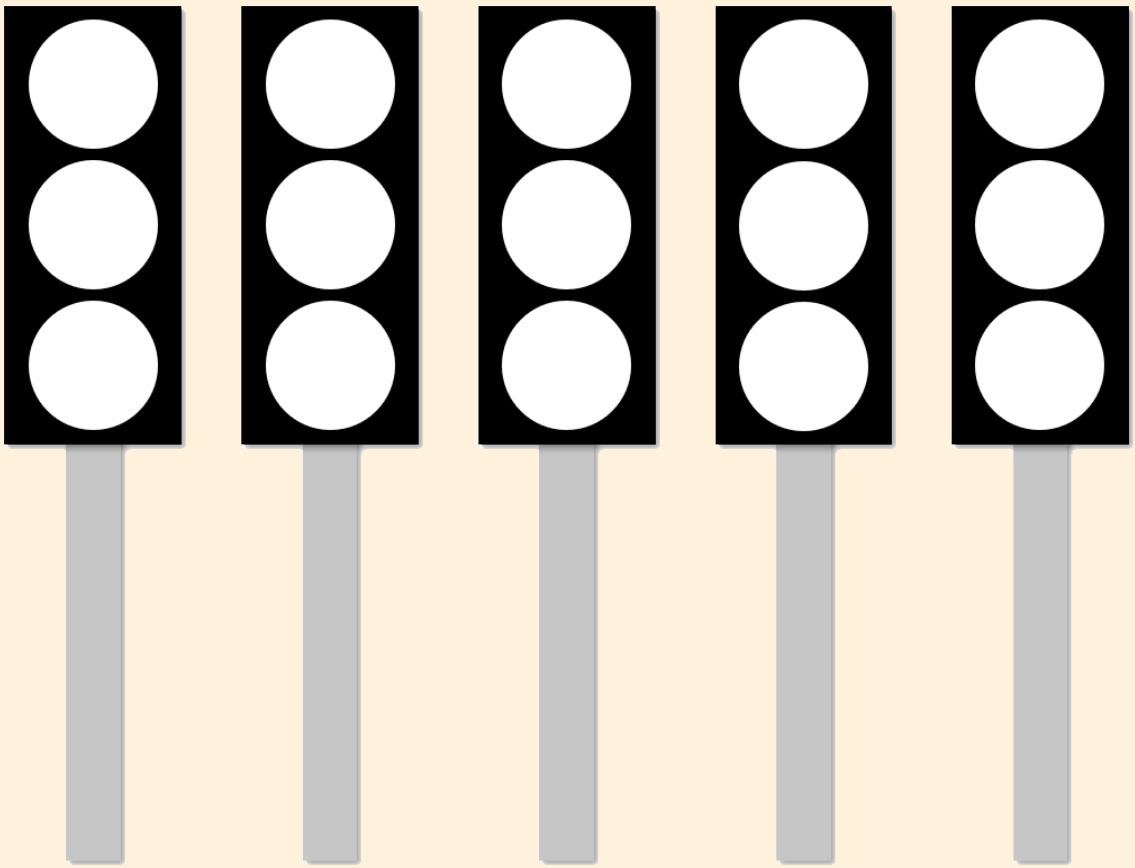
ZIEL DER STATION

Ampeln begegnen dir im Alltag ständig. Und zwar egal ob du im Bus, im Auto, mit dem Fahrrad oder auch zu Fuß unterwegs bist. Je nach Größe der Kreuzung kann so eine Ampelschaltung sehr schnell sehr kompliziert werden, eine einfache Ampel kannst du allerdings ganz einfach selbst realisieren.

Wir wollen uns daher nun damit beschäftigen, eine Ampelschaltung mit drei verschieden farbigen LEDs, sprich rot-gelb-grün, zu realisieren. Dazu muss man natürlich erstmal wissen, welche Ampelphasen es überhaupt gibt.

AUFGABE 1 – AMPELPHASEN

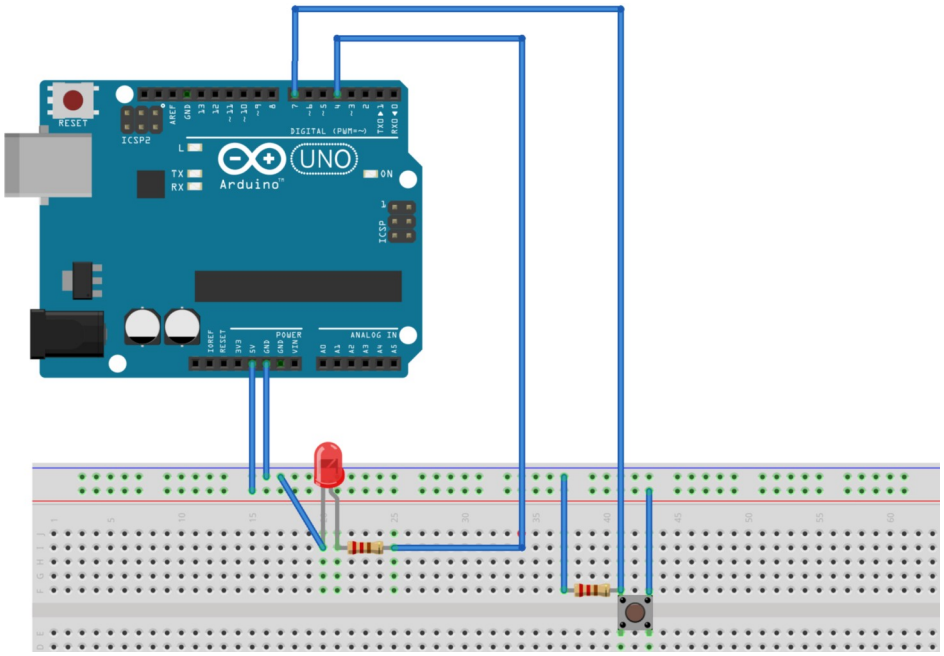
Wir unterscheiden insgesamt 5 verschiedene Ampelphasen. Zeichne diese in der nachstehenden Grafik mit den richtigen Farben ein.



WIEDERHOLUNG - LED STEUERN

Zunächst starten wir mit einer kleinen Wiederholung und schauen uns nochmal an, wie man eine LED ein-und ausschaltet nach einer Wartezeit, sowie mit Hilfe eines Tasters.

Dazu benötigen wir zunächst die dazugehörige Schaltung. Beachte dabei, dass vor allem die LED unbedingt mit einem 230 Ω Widerstand verbunden werden muss. Wie dies aussiehst, kann du in der nachfolgenden Grafik nachvollziehen.



AUFGABE 1 – REALISIERUNG DER SCHALTUNG

Schaffe zunächst die Grundlage für unser Programm und realisiere die Schaltung zunächst mit einer LED und einem Taster.

Wie geht es weiter? Wir legen uns Variablen fest, welche stellvertretend für den digitalen Pin stehen, an denen LED und Taster angeschlossen sind. Dies sieht im Programmcode wie folgt aus:

```
int roteLed = 4;

int taster = 7;
```

Jetzt folgt die Festlegung des Pin-Modus. Die LED definieren wir als `OUTPUT` und den Taster natürlich als `INPUT`, da wir dessen Status auslesen wollen.

```
pinMode(PIN, MODUS);
```

Diese Definition muss innerhalb der `setup()`-Methode, zu Beginn des Sketches geschehen. Nun wollen wir die LED blinken lassen. Dazu haben wir ja bereits die `delay()`-Funktion kennengelernt, welche die Ausführung des Programms für eine Sekunden passiert. Weiterhin kannst du festlegen, ob am digitalen Pin `HIGH` oder `LOW` anliegt. Nutze dazu die folgende Codezeile:

```
digitalWrite(PIN, HIGH oder LOW);
```

AUFGABE 2 – LASSE DIE LED BLINKEN

Initialisiere die Variablen und definiere den Pin-Modus. Nutze dann die `delay()`- und `digitalWrite()`-Funktion so, dass die LED blinkt.

Weiter geht es mit dem Ein- und Ausschalten der LED mit Hilfe eines Tasters. Dazu müssen wir zunächst den Status des Taster auslesen, sprich ob nun `LOW` oder `HIGH` am digitalen Pin anliegt. Wird der Taster gedrückt liegt am digitalen Pin `HIGH` an, andernfalls `LOW`. Um dies auszulesen nutzen wir die folgende Funktion:

```
digitalRead(PIN);
```

Nach dem wir wissen, ob `HIGH` oder `LOW` am Pin anliegt, müssen wir mit Hilfe der einfachen Verzweigung noch entscheiden was unter welcher Bedingung passieren soll. Verbalisiert ergibt sich:

WENN Taster gedrückt **DANN** LED an **SONST** LED aus

Im Programmcode, unter Nutzung der eben benannten Funktion und des Vergleichsoperators == sieht dies folgendermaßen aus.

```
if(digitalRead(PIN)==HIGH){
      digitalWrite(PIN, HIGH);
} else{
      digitalWrite(PIN, LOW);
}
```

AUFGABE 3 – LED MIT TASTER STEUERN

Setze die einfache Verzweigung in deinem Programmcode um und steuere die LED mit dem Taster.

WEITER ZUR AMPELSCHALTUNG

Die Grundlagen sind wiederholt, jetzt können wir uns mit unserer Ampelschaltung auseinandersetzen. Wie sieht das ganze aus? Die fünf Ampelphasen hast du ja weiter vorn bereits aufgemalt. Dazu gilt es weiterhin noch zu bedenken, dass die Phasen nicht alle die gleiche Dauer haben. So dauern die rot und grün Phasen deutlich länger als die Zwischenphasen.

In einem ersten Schritt nutzt du die Möglichkeit das Verhalten der LED mit der `delay()`-Funktion zu steuern und schreibst immer Codeblöcke für eine LED. Beispielhaft für die erste, rein grüne, Ampelphase, kann das so aussehen:

```
digitalWrite(roteLed, HIGH);
digitalWrite(gelbeLed, LOW);
digitalWrite(grueneLed, HIGH);

delay(2000);
```

AUFGABE 4 – DIE AMPELSCHALTUNG

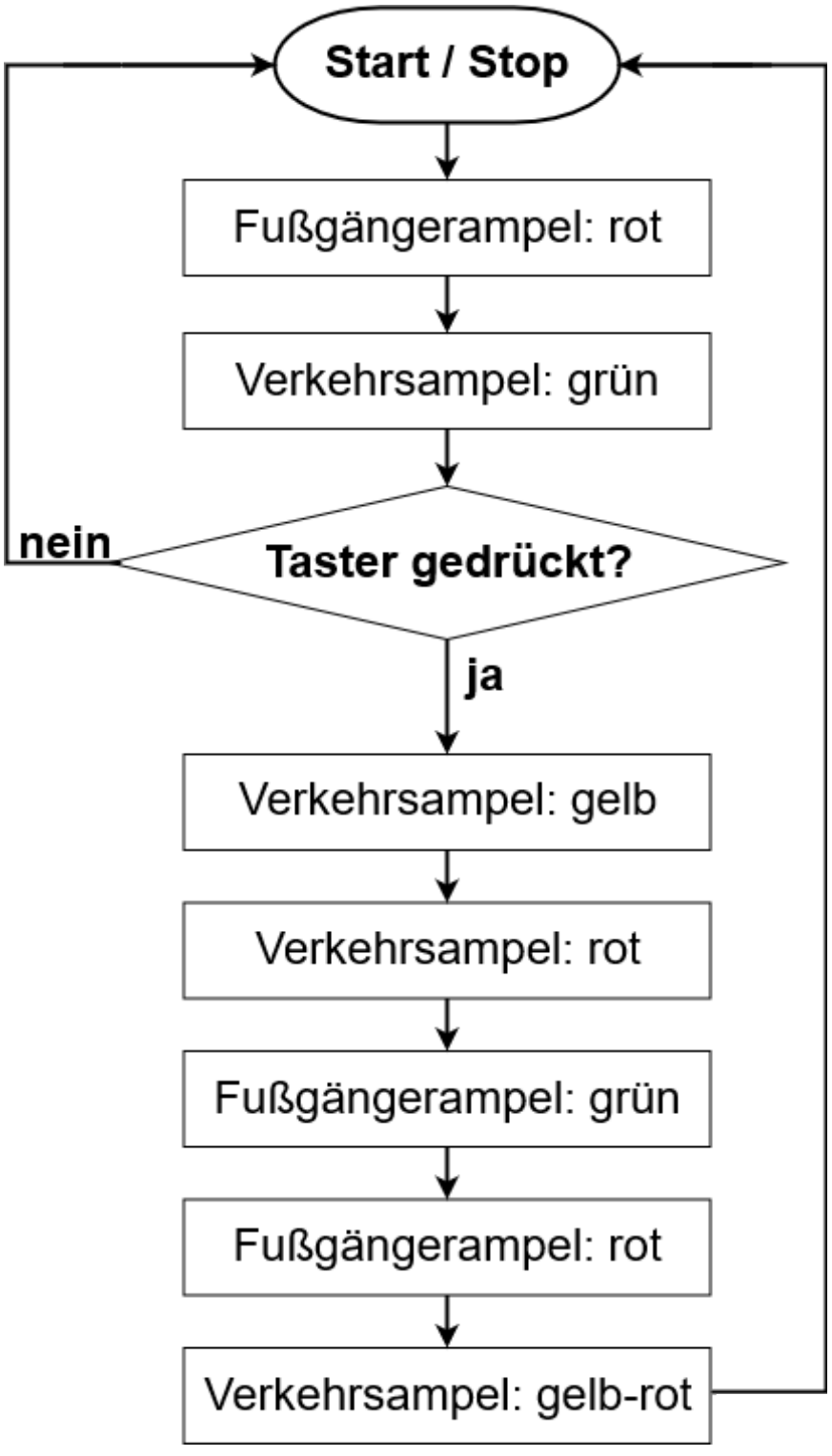
Ergänze dein bisherigen Projekt um zwei weitere LEDs, so dass du die Farben rot, grün und gelb darstellen kannst. Vergiss dabei nicht, für jede LED den Pin-Modus festzulegen. Im Anschluss gestaltest du, wie oben bereits gezeigt, alle weiteren Ampelphasen. Teste im Anschluss die Abfolge der Phasen und variiere die Wartezeiten zwischen den Phasenwechseln so, dass diese bestmöglich die Realität abbilden. Mach es dir so einfach wie möglich und nutze copy & paste. Dann musst du jeweils nur noch `HIGH` oder `LOW` anpassen.

UND WAS IST MIT DEN FUSSGÄNGERN?

Darum kümmern wir uns jetzt! Vielleicht erschließt sich dir auch genau in diesem Moment, warum du dich zur Wiederholung auch mit dem Taster beschäftigt hast. Denn wir wollen unserer Ampelschaltung noch die Möglichkeit hinzufügen, dass Fußgänger einen Taster betätigen können, um sicher über die Straße gehen zu können.

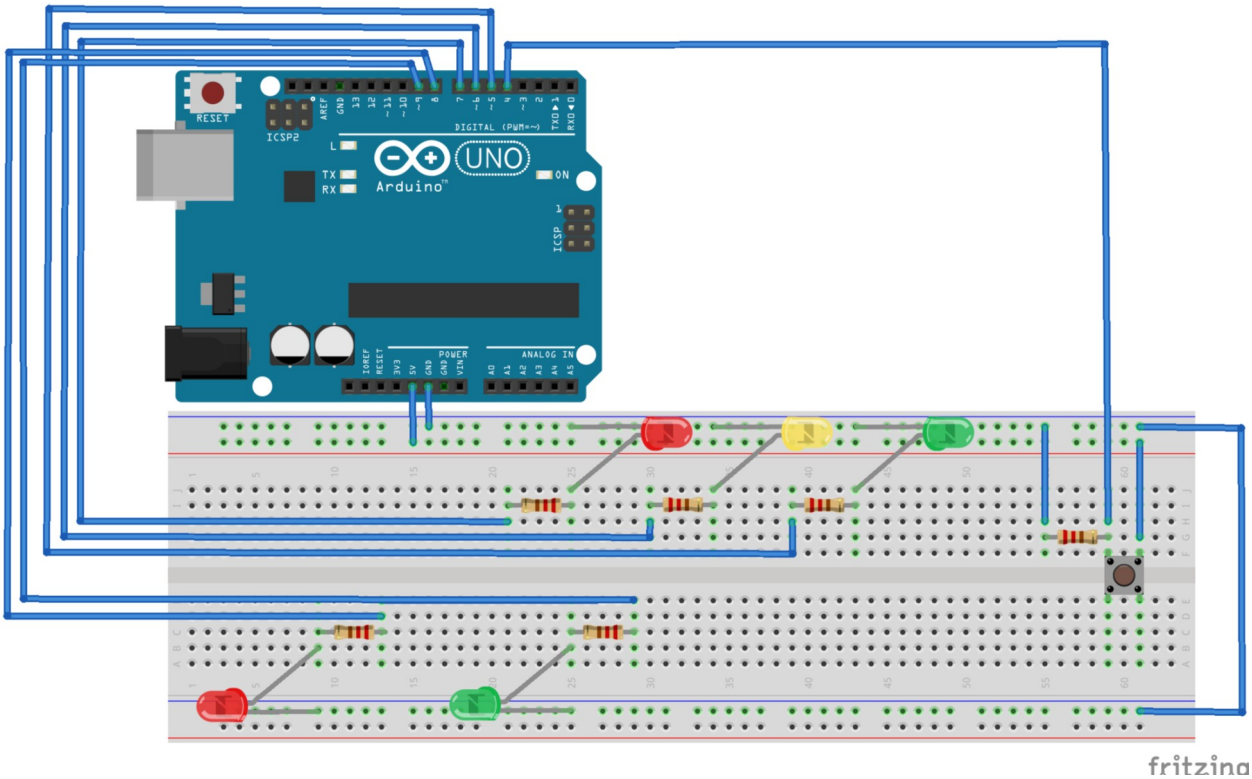
PROGRAMMLOGIK

Das nächste Bild zeigt dir die Logik des nachfolgenden Programms. Zunächst soll dauerhaft die Fußgängerampel rot und die Verkehrsampel grün sein. Erst wenn der Taster gedrückt wurde, sollen die anderen Ampelphasen abgearbeitet werden.



ERWEITERUNG DER SCHALTUNG

Wie du bestimmt bereits gemerkt hast, reichen die jetzigen drei LEDs natürlich nicht mehr aus. Wir müssen also unsere Schaltung erweitern. Dazu kannst du dich am unten abgebildeten Aufbau orientieren.



AUFGABE 5 – SCHALTUNG ERWEITERN

Erweitere deine Schaltung um die zwei zusätzlich notwendigen LEDs. Initialisiere die dazugehörigen Variablen und lege den Pin-Modus auf `OUTPUT` fest.

UMSETZUNG IN PROGRAMMCODE

Wir beginnen damit, innerhalb der `loop()`-Methode die notwendigen Befehle für eine grüne Verkehrs- und für eine rote Fußgängerampel zu programmieren. Nach dem die LEDs alle farblich richtig gesetzt wurden, wollen wir überprüfen ob der Taster gedrückt wurde um gegebenenfalls die anderen Ampelphasen darzustellen. Zu diesem Zweck findet auch hier die einfache Verzweigung Anwendung.

```
void loop() {
  digitalWrite(roteLed, HIGH);
  digitalWrite(gelbeLed, LOW);
  digitalWrite(grueneLed, LOW);
  digitalWrite(fussgaengerroteLed, LOW);
  digitalWrite(fussgaengergrueneLed, HIGH);
  if (digitalRead(button) == HIGH) {
     ...
  }
}
```

AUFGABE 6 – AMPEL FÜR FUSSGÄNGER

Mit Hilfe des obigen Codes hast du auf jeden Fall schon mal eine Ampel, welche dem Verkehr immer grünes Licht gibt. Die Verzweigung ist ja auch bereits vorgegeben, somit musst du jetzt nur noch die Ampelphasen in der Verzweigung einfügen.

Im Anschluss gilt es dein Programm zu testen und natürlich weiter zu optimieren, vor allem hinsichtlich der Wartezeiten zwischen den einzelnen Phasen.

DU HAST NOCH NICHT GENUG VON DIESER STATION?

Kein Problem - Frage die Betreuer einfach nach dem Zusatzblatt! Dort wirst du deiner Ampel noch einen akustischen Warnton hinzufügen, um selbstverständlich auch Gehörlose sicher über die Straße führen zu können.

Grafik auf dem Deckblatt: *Ampel, Clker-Free-Vector-Images, Pixabay License, https://pixabay.com/de/service/license/, https://pixabay.com/de/vectors/ampel-rot-schwarz-gr%C3%BCn-gelb-24177/*
Screenshots: *fritzing electronics made by easy und Arduino IDE 1.8.12 (windows)*
Alle weiteren Grafiken: *Patrick Binkert, EduInf@TUD*