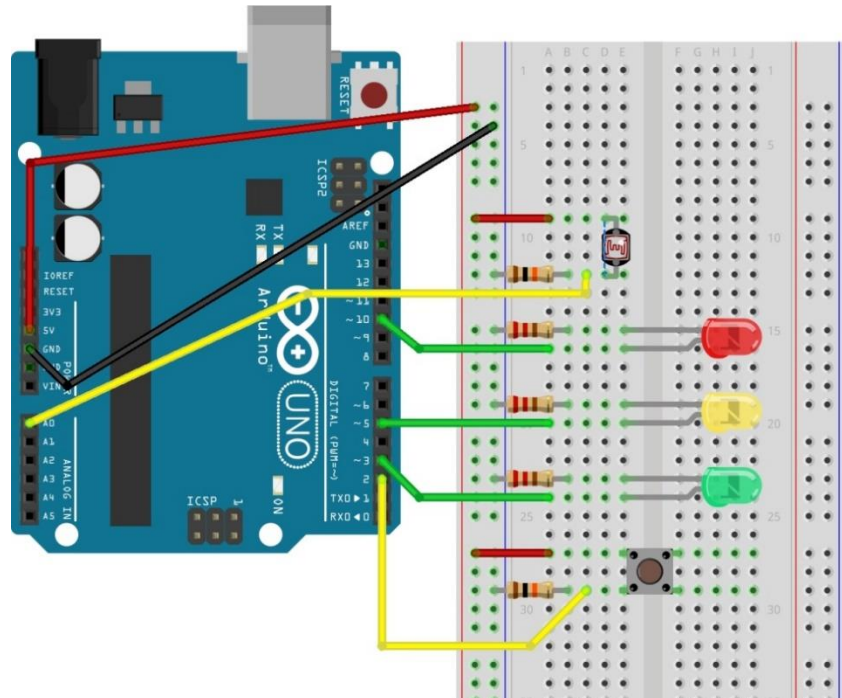


Arduino – eigene Befehle programmieren

Du verwendest bei diesem Arbeitsblatt weiterhin die Arduino-Steckbrett-Kombination, auf der die folgende Schaltung zur Programmierung einer Ampelsteuerung oder eines Morseapparats vorbereitet ist:



Aufgabe 1) Programme nun deinen ersten „eigenen“ Arduino-Befehl. Durch Aufrufen dieses Befehls im `loop`-Block des Arduino-Programms soll die grüne Leuchtdiode dreimal blinken.

Nenne diesen Befehl `greenLed3Blink` und füge diese **Befehlsüberschrift** und das **geschwungene Klammerpaar** als Rahmen für den zu codierenden Programmblock wie in nebenstehender Abbildung gezeigt zwischen den **Variablendeklarationen** und dem `setup`-Programmblock ein.

```
int redLedPin = 10;
int yellowLedPin = 5;
int greenLedPin = 3;
```

```
void greenLed3Blink() {
}
```



Zwischen die geschwungenen Klammern kommen nun die Befehle, die das dreimalige Blinken der grünen Leuchtdiode bewirken – verwende dazu den Code, den du beim Lösen der **Aufgabe 3)** von Aufgabenblatt **ST_AA_05** geschrieben hast.

```
void setup() {
  pinMode(redLedPin, OUTPUT);
  pinMode(yellowLedPin, OUTPUT);
  pinMode(greenLedPin, OUTPUT);
  //set initial LED-states
}

void loop() {
  delay(5000);
  digitalWrite(redLedPin, LOW);
  digitalWrite(yellowLedPin, HIGH);
  delay(2000);
  digitalWrite(greenLedPin, HIGH);
  digitalWrite(yellowLedPin, LOW);
  delay(5000);
  greenLED3Blink();
  digitalWrite(yellowLedPin, HIGH);
}
```

Teste sodann diesen Befehl, indem du den Befehl im `loop`-Block des Arduino-Programms anstelle des ursprünglichen „Blink-Codes“ verwendest:

Aufgabe 2) Füge nun deinem Arduino-Projekt wie in nebenstehender Abbildung angedeutet zwei weitere Befehle namens `yellowLed3Blink` bzw. `redLed3Blink` hinzu, nach deren Aufruf im `loop`-Block des Arduino-Programms die gelbe bzw. die rote Leuchtdiode zu blinken beginnen.

Achte darauf, dass der Block für den Code eines Befehls mit einer geschwungenen Klammer geschlossen wird,  bevor du die Überschrift des nächsten Befehls schreibst! 

```
int redLedPin = 10;
int yellowLedPin = 5;
int greenLedPin = 3;

void greenLed3Blink() {
    ...
}

void yellowLed3Blink() {
    ...
}

void redLed3Blink() {
    ...
}

void setup() {
    pinMode(redLedPin, OUTPUT);
    pinMode(yellowLedPin, OUTPUT);
    pinMode(greenLedPin, OUTPUT);
    //set initial LED-states
}
```

Teste diese neuen Befehle auch!

Aufgabe 3) Vermutlich ist dir beim Lösen der Aufgabe 2) aufgefallen, dass der Code der Befehle für dreimaliges Blinken der grünen, der gelben oder der roten Leuchtdiode ziemlich ähnlich ist – es ist lediglich der Befehlsname zu ändern und die Variable `greenLedPin` ist durch `yellowLedPin` oder `redLedPin` zu ersetzen – richtig?

Genau in einer solchen Situation werden die Daten, die ein Befehl „von außen“ erhalten kann, wichtig. In der Informationsdatei [ST_I_06](#) ist erklärt, dass bei einem Befehl mit der **Befehlsüberschrift**

```
void led3Blink(int pin)
```

im **Befehlscode** `pin` wie eine Variable (d.h. wie vorher `redLedPin`, `greenLedPin` oder `yellowLedPin`) verwendet werden kann. Der Vorteil eines solchen Befehls liegt nun darin, dass du beim Aufrufen des Befehls steuern kannst, ob die rote Leuchtdiode (`led3Blink(redLedPin);`)

oder die grüne Leuchtdiode (`led3Blink(greenLedPin);`)

oder die gelbe Leuchtdiode (`led3Blink(yellowLedPin);`)

dreimal blinken soll.

Füge nun also deinem Arduino-Projekt diesen Befehl `led3Blink(int pin)` hinzu und teste diesen neuen Befehl.

Als zweite Anwendung der vorbereiteten Schaltung hast du im Rahmen von Aufgabenblatt [ST_AA_05](#) auch einen **Morseapparat** programmiert. Auch da kann durch Programmieren eigener Befehle einiges kürzer codiert werden – damit beschäftigst du dich in den folgenden Aufgaben:

Aufgabe 4) In Aufgabe 6) vom Aufgabenblatt **ST_AA_05** hast du alle drei Leuchtdioden der vorbereiteten Schaltung verwendet, um längere Nachrichten im Morsecode übertragen zu können: Eine Leuchtdiode für das „Senden“ der kurzen und langen Morsesignale, eine Leuchtdiode um anzuzeigen, wann ein Buchstabe vollständig „gesendet“ wurde, und die dritte Leuchtdiode für die Anzeige eines Wortendes.

Wiederhole zunächst deine Überlegungen zur Lösung dieser Aufgabe 6), indem du ein Arduino-Programm zum Morsen des kurzen Satzes „Das passt“ programmierst.

Aufgabe 5) Dir ist beim Lösen von Aufgabe 4) sicherlich aufgefallen: Bereits für einen kurzen Satz wie „Das passt“ wird der Programmcode recht lang (und das Schreiben des Programmcodes recht langweilig). Immer und immer wieder sind die Codestücke für „kurz“ und „lang“ einzutippen! Da können selbst programmierte neue Befehle Abhilfe schaffen:

Der Befehlscode für das „kurze“ Morsesignal lässt sich umgangssprachlich so:

```
LED einschalten
kurz warten
LED ausschalten
warten
```

und jener für das „lange“ Morsesignal so:

```
LED einschalten
lang warten
LED ausschalten
warten
```

beschreiben.

Füge dem Morseprogramm zunächst zwei Befehle

```
void shortSignal(int pin) {           und void longSignal(int pin) {
    ...                               }
}
```

hinzu. In der Klammer soll die Nummer des Kontaktes jener Leuchtdiode übergeben werden, die für das Morsen verwendet wird. Beispielsweise könnte also der neue Befehl `shortSignal` im `loop`-Programmblock durch `shortSignal(redLedPin)`; aufgerufen werden.

Teste die beiden neuen Befehle, indem du den Satz „Das passt.“ nun mit Hilfe dieser Befehle morst...

Aufgabe 6) Die Befehle zum Senden eines kurzen bzw. eines langen Morsesignals können leicht zu Befehlen für die einzelnen Zeichen des Alphabets kombiniert werden. Der Code für den Buchstaben „S“ könnte zum Beispiel so aussehen:

```
void letterS(int pin) {
    shortSignal(pin);
    shortSignal(pin);
    shortSignal(pin);
}
```

und im `loop`-Block einfach als `letterS(redLedPin)`; aufgerufen werden...

Füge deinem Morseprogramm nun noch neue Befehle für die Buchstaben D, A, S, P und T hinzu und morse nun den Satz „Das passt“ mit Hilfe dieser neu dazuprogrammierten Befehle.

Du merkst wohl: Durch **selbst programmierte Befehle** lässt sich das Programmieren ziemlich **vereinfachen**, oder?