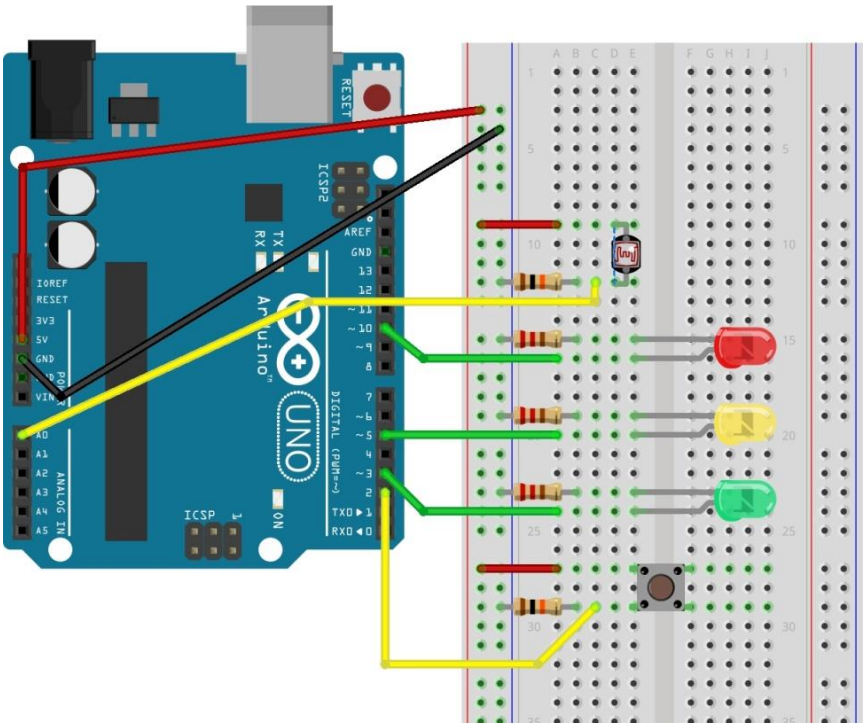
Arbeiten in der Arduino-Programmierumgebung

An deinem Arbeitsplatz findest du eine Arduino-Steckbrett-Kombination, auf der die folgende Schaltung vorbereitet ist:



Aufgabe 1) Das nebenstehende Programm bringt bei der „neuen“ Schaltung die gelbe Leuchtdiode (LED) zum Blinken. Schreib es ab und teste es (das heißt: Übertrage es auf den Arduino)!
Neu ist an diesem Programm die Zeile

```
//set initial LED-state
```

Die zwei Schrägstriche am Beginn der Zeile zeigen an, dass es sich um einen sogenannten **Kommentar** handelt. Das ist ein Programmteil, der beim Ausführen des Programms vom Arduino-Board nicht beachtet wird, für uns Menschen aber segensreich sein kann, weil dadurch Teile des Programms erklärt werden können.

```
int yellowLedPin = 5;

void setup() {
  pinMode(yellowLedPin,OUTPUT);
  //set inital LED-state
  digitalWrite(yellowLedPin,LOW);
}

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

Aufgabe 2) Erweitere nun zunächst den `setup`-Programmteil so, dass auch für die beiden anderen Leuchtdioden zwei Variable namens **`redLedPin`** bzw. **`greenLedPin`** vereinbart werden. Dazu musst Du herausfinden, welche Kontakte in der Schaltung für diese Leuchtdioden verwendet werden.
Zudem soll zu Beginn die rote Leuchtdiode leuchten, die beiden anderen sollen dunkel sein.

Verändere dann den `loop`-Programmteil so, dass die Ampel von „rot“ über „gelb“ auf „grün“ und dann wieder über „gelb“ auf „rot“ umschaltet! Teste dein Programm!

Aufgabe 3) Üblicherweise blinkt bei einer Ampel das grüne Licht dreimal, bevor die Ampel auf „gelb“ umschaltet. Erweitere den `loop`-Programmteil in deinem Programm aus Aufgabe 1) um diese Funktionalität – d.h. die Reihenfolge soll nun „rot“ – „gelb“ – „grün“ – „grün blinkend“ – „gelb“ (und dann wieder „rot“ – …) sein und teste das Programm.

Mit der vorbereiteten Schaltung kann aber nicht nur eine Ampelsteuerung programmiert werden, wir können damit auch mit Hilfe des Morsecodes Nachrichten übermitteln. Falls du mit den Begriffen „morsen“, „Morsealphabet“ oder „Morseapparat“ gar nichts anfangen kannst, dann lies bitte das Informationsblatt zum Codieren durch und löse dann die zwei Aufgaben zum Morsen. Wenn du das getan hast, oder wenn du über das Morsen ohnehin Bescheid weißt, dann widme dich bitte den folgenden Aufgaben:

Aufgabe 4) Für einen optischen Morseapparat benötigen wir eigentlich nur eine einzige Leuchtdiode, die die Morsesignale als Abfolge von „kurz“ und „lang“ sendet. Du kannst dir also aussuchen, welche der drei LEDs in der Ampelschaltung für die Übertragung der Morsesignale verwendet werden soll. Die „Ampelschaltung“ wird dann zur „Morseapparat-Schaltung“. Wenn Du Dich entschieden hast…

…schreibe ein Arduino-Programm, das deinen Arduino-Morseapparat (fortwährend) SOS senden lässt (zur Erinnerung: Der Morsecode für das Hilfesignal SOS ist dreimal „kurz“ – dreimal „lang“ – dreimal „kurz“). Teste dein Programm!

Aufgabe 5) Vielleicht ist dir beim SOS-Morsen mit dem gerade eben geschriebenen Programm aufgefallen, dass der gesendete Morsecode dreimal „kurz“ – dreimal „lang“ – dreimal „kurz“ auch anders, z.B. als zweimal „kurz“ – „kurz“ und „lang“ – „lang“ – „lang“ und dreimal „kurz“, verstanden werden kann – und das bedeutet dann nicht SOS sondern IATB. Das Problem besteht offenbar darin, dass nicht klar ist, wann der Morsecode für ein Zeichen endet und der für das nächste Zeichen beginnt.

a) Suche nach weiteren Möglichkeiten, wie der Morsecode dreimal „kurz“, – dreimal „lang“ – dreimal „kurz“ von einem Empfänger falsch verstanden werden könnte.

b) Das oben beschriebene Problem lässt sich leicht lösen: Es gibt ja noch zwei ungenutzte LEDs im „Ampelschaltungsmorseapparat“. Wähle eine der beiden Leuchtdioden, die nicht zum Morsen verwendet wird, und verändere das Programm aus Aufgabe 4) so, dass diese LED immer dann aufleuchtet, wenn der Code für ein Zeichen vollständig übertragen wurde. Teste dieses neue Programm!

Aufgabe 6) Ein Empfänger der Morsenachricht, die mit dem in Aufgabe 5) verbesserten Morseprogramm gesendet wurde, kann diese nun eindeutig zu S O S S O S S O S S O S … decodieren. Das muss aber noch nicht bedeuten, dass er die Zeichenfolge richtig zu „Worten“ zusammenfasst. Vielleicht wundert sich der – zugegebenermaßen etwas einfältige – Empfänger ja, dass der Sender SO SSO SSO SSO SSO … sendet, also immer das Wort „so“, ab dem zweiten Wort dazu noch falsch geschrieben! Es wäre also hilfreich, wenn ein weiteres Signal anzeigt, dass ein Wort vollständig übertragen wurde!

Verwende nun auch noch die dritte der Leuchtdioden und erweitere das Morseprogramm so, dass diese LED immer dann aufleuchtet, wenn ein Wort vollständig übertragen wurde. Teste dieses neue Programm!

Aufgabe 7) Die folgenden Aufgabenstellungen dienen dazu, die Modellvorstellung über Variable zu festigen – bearbeite bitte alle drei Aufgabenteile:

a) Vervollständige den nachfolgenden Text, indem du jeweils den passenden Begriff bzw. die passenden Worte an der dafür vorgesehenen Stelle einsetzt:

Bei der Variablendeklaration wird für _____ ein passender Speicherbereich reserviert. In diesem Speicherbereich ist zunächst _____ gespeichert. Der Inhalt eines _____ lässt sich als Folge von _____ und/oder _____ darstellen.

…verwende dabei Worte/Begriffe aus der folgenden Liste:
die Zahl Null, Nullen, den Computer, die Variable, eine dezimale Zahl, Einsen, Speicherbereichs, ein unbekanntes Datum, die Zahl Fünf, Bildschirms.

b) Die grüne Leuchtdiode wird in der verwendeten Schaltung über den Steckkontakt mit der Nummer ~3 angesteuert. Dies soll im Programm mit Hilfe einer Variablen namens `greenLedPin` erfolgen. Dazu wird mit dem Befehl `int greenLedPin;` ein passender Speicherbereich reserviert – wir sprechen von einer **Variablendeklaration**.

Nach dieser Variablendeklaration steht im reservierten Speicherbereich (als Bitmuster codiert)

❑ immer die Zahl 0.

❑ immer die Zahl 3.

❑ irgendeine Zahl, man kann nicht sagen, welche.

❑ gar nichts, weil der Speicher noch leer ist.

Markiere die richtige(n) Aussage(n).

c) Mit dem Befehl `greenLedPin = 3;` wird in den für die Variable namens `greenLedPin` reservierten Speicherbereich die Zahl 3 geschrieben.

Beim Ausführen des Befehls `digitalWrite(greenLedPin, HIGH);` wird dieser Wert im Speicher gelesen.

Nach dem Lesen dieses Wertes steht in dem für die Variable namens `greenLedPin` reservierten Speicherbereich

❑ noch immer die Zahl 3.

❑ die Zahl 0.

❑ irgendeine Zahl, man kann nicht sagen, welche.

❑ gar nichts, weil der Speicherinhalt gelesen wurde und der Speicher nun leer ist.

Markiere die richtige(n) Aussage(n).

Bonusaufgabe: Auf Seite drei der Informationsdatei (ST_I_05Arduino_Programmierumgebung) findest du zwei vereinfachte Darstellungen von Bereichen im Speicher des Computers. In der ersten dieser Abbildungen sind die Inhalte dieser Speicherbereiche als Bitmuster dargestellt, in der zweiten als entsprechende ganze dezimale Zahlen.

Überprüfe an mindestens drei dieser Speicherinhalte nachvollziehbar, ob die Umwandlung des Bitmusters in die entsprechende ganze Zahl korrekt ist – z.B. so:

$10010011_2 = 1 \cdot \mathbf{128} + 0 \cdot \mathbf{64} + 0 \cdot \mathbf{32} + 1 \cdot \mathbf{16} + 0 \cdot \mathbf{8} + 0 \cdot \mathbf{4} + 1 \cdot \mathbf{2} + 1 \cdot \mathbf{1} = 128 + 16 + 3 = 147_{10}$