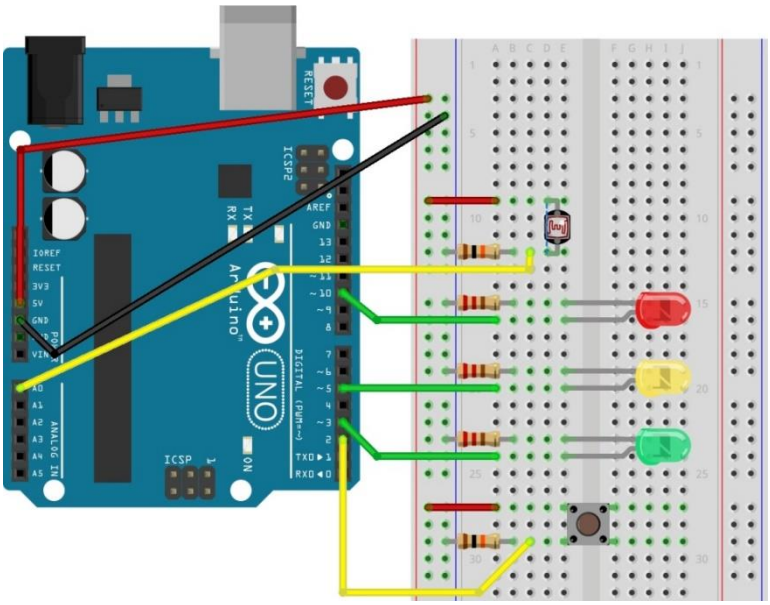
Arduino – ein Programm trifft Entscheidungen

Du verwendest bei diesem Arbeitsblatt zunächst weiterhin die Arduino-Steckbrett-Kombination, auf der die Schaltung zur Programmierung einer Ampelschaltung oder eines Morseapparats vorbereitet ist.



Arbeitsanregungen zur Schalterprogrammierung

In der Informationsdatei namens **ST_I_08_Arduino_Verzweigungen** ist auf den ersten beiden Seiten erklärt, wie im Programm entschieden werden kann, ob der in die Schaltung eingebaute Druckknopf-Schalter gedrückt wurde, sodass nach dem Drücken des Schalters anderer (man sagt auch: „alternativer“) Programmcode ausgeführt wird. Studiere diese Informationsdatei und bearbeite daraufhin die folgenden Aufgaben:

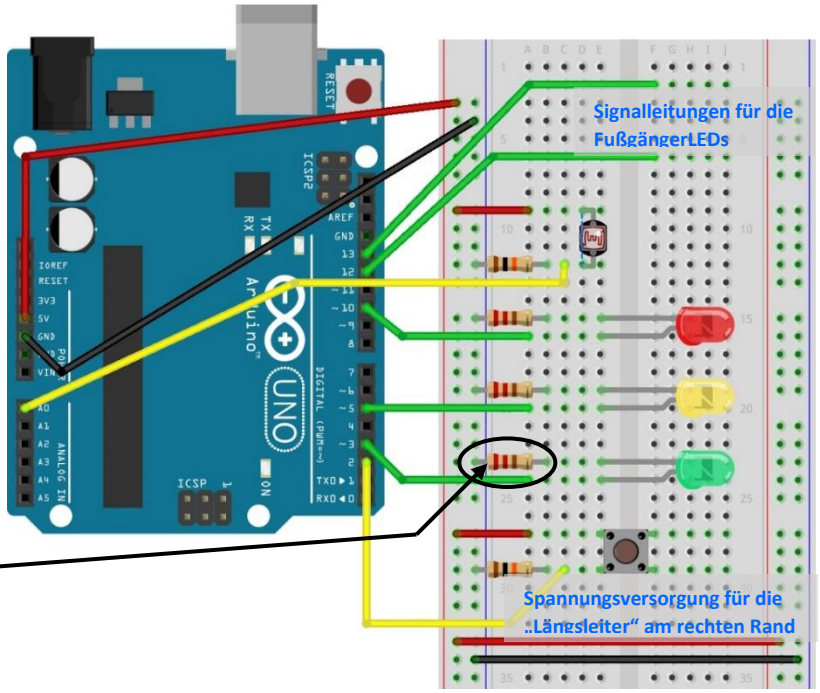
Aufgabe 1) Bei Ampelanlagen treten Druckknöpfe dann auf, wenn eine Ampel für den fließenden Verkehr (auf der Straße) mit einer Fußgänger- (oder Radfahrer-) Ampel kombiniert ist. In diesem Fall zeigt die Ampel für den fließenden Verkehr zunächst „grün“ und schaltet nach Drücken eines Druckknopfes über „grün blinkend“ und „gelb“ auf „rot“ (z.B. damit Fußgänger die Fahrbahn überqueren können), und nach einiger Zeit dann wieder über „gelb“ auf „grün“.

Modifiziere dein bisheriges Ampelsteuerungs-Programm und ergänze es um die „Druckknopf“-Befehle, sodass das Programm die Ampelschaltung so steuert, wie es im ersten Absatz dieser Aufgabenformulierung beschrieben ist.

Aufgabe 2) Bisher hast du eine Ampel mit drei Farben programmiert – bei uns sind solche Ampeln für den fließenden Verkehr üblich. In der letzten Aufgabe ist aber schon angeklungen, dass die Kombination einer druckknopfgesteuerten Dreifarbenampel für den fließenden Verkehr mit einem Signalgeber für Fußgänger kombiniert werden könnte. In Österreich haben solche Fußgängerampeln meist zwei Farben: rot und grün.

Erweitere dazu zunächst die Ampelschaltung um zusätzlich eine rote und eine grüne Leuchtdiode. Diese Leuchtdioden müssen wieder über Signale vom Arduino ein- und ausgeschaltet werden können, d.h. das jeweils längere „Beinchen“ (die **Anode**) **der LEDs** ist jeweils mit einem digitalen Kontakt auf dem Arduino-Board zu verbinden. Der Übersichtlichkeit halber bieten sich die **Kontakte mit den Nummern 12 und 13** an.

Dabei ist zu bedenken, dass die „linke Seite“ der Steckplatine schon ziemlich voll ist, sodass die beiden neuen („Fußgänger“-) Leuchtdioden sinnvollerweise in der „rechten Hälfte“ platziert werden sollten. Am einfachsten geht das, wenn auch die „Längsleitungen“ am rechten Rand geeignet elektrische Energie zur Verfügung stellen können – die Erweiterung der Ampelschaltung sieht daher im Prinzip so aus:



Modifiziere deine Ampelschaltung entsprechend und füge die fehlenden „Fußgänger“-LEDs hinzu. Vergiss nicht, dass deren „kurze Beinchen“ über einen Widerstand derselben Art wie jene LEDs, die schon in die Schaltung eingebaut sind, mit „Erde“ („**GND**“) verbunden werden müssen, und lass‘ deine Schaltung bitte kontrollieren!

Wenn du das Programm aus Aufgabe 1) wiederverwendest, ist nun nur noch eine kleine Programmänderung notwendig: Nach dem Knopfdruck schaltet beim bisherigen Programm die Fahrzeugampel auf „rot“. Wenn die Fahrzeugampel „rot“ zeigt, soll die Fußgängerampel klarerweise „grün“ zeigen, wenn die Fahrzeugampel wieder auf „grün“ geschaltet hat, muss die Fußgängerampel auf „rot“ sein. Zusätzlich könnte die Fußgängerampel auch noch „grün“ blinken, bevor sie auf „rot“ umschaltet…

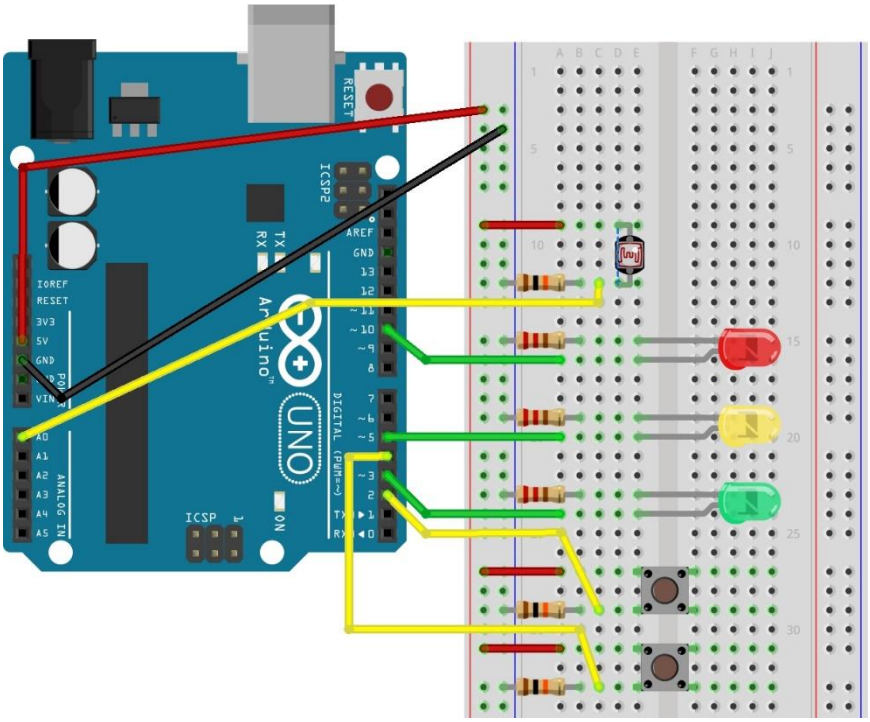
…falls du unsicher bist, was dazu zu machen ist – ein paar Tipps:

Erweitere zunächst den `setup`-Programmteil so, dass auch zwei Variablen namens **redPLedPin** und **greenPLedPin** für die beiden Fußgänger-Leuchtdioden vereinbart und in den passenden „Modus“ gesetzt werden („P“ kommt vom englischen Wort für Fußgänger: pedestrian). Nachdem die Ampel für den fließenden Verkehr vor dem Betätigen des Druckknopfes „grün“ zeigt, soll die rote Fußgänger-LED zu Beginn leuchten, die grüne aber dunkel sein. Das Umschalten nach dem Drücken des Druckknopfes muss dann noch im `loop`-Programmteil zu codieren…

Hinweis: Falls du gleich den Photowiderstand Deiner Ampelschaltung programmieren willst, beschäftige dich als Nächstes mit Aufgabe 4 auf der übernächsten Seite,

…ansonsten geht's weiter mit:

Aufgabe 3) Mit Hilfe von Drucktastern, von denen du einen in den vorangehenden Aufgaben schon verwendet und programmiert hast, um die Ampelschaltung „händisch zu steuern", lässt sich auch das Arduino-Programm zur Steuerung des „Morse-Apparats" erweitern (im Sinne von: „flexibler gestalten"):



Erweitere deine „optische Morseapparatschaltung" folgendermaßen um einen weiteren Druckknopf:

Beachte, dass du einen Widerstand mit den Farbringen braun – schwarz – orange einbauen musst!

Beim Drücken des einen Druckknopfes soll ein kurzes Morsesignal gesendet werden, beim Drücken des anderen ein langes Morsesignal, sodass du beliebige Wörter morsen kannst!

Codiere ein entsprechendes Programm – sende dann einer anderen Person Morsebotschaften, die von dieser „entschlüsselt" werden sollen.

Oder findest du auch eine andere Person, die Dir „geheime" Morsebotschaften schickt?

Vermutlich stellt Ihr beim Morsen recht schnell fest, dass nachdem ein Buchstabe vollständig gesendet wurde, etwas länger gewartet werden sollte, damit der Empfänger das Gesendete richtig decodieren kann.
Alternativ könnte eine zweite der drei LEDs verwendet werden, die anzeigt, wann ein Buchstabe vollständig gemorst wurde. Da könnte man also einen dritten Druckknopf einbauen: Wenn dieser dritte Druckknopf betätigt wird, soll eine weitere LED aufleuchten – die Person, die die Nachricht empfängt, weiß dann, dass ein Buchstabe vollständig übertragen wurde.

Mit dem bereits erworbenen Vorwissen schaffst du es sicher, ein solches Programm zu codieren! „Experimentiere" dann auch ein bisschen damit ☺.

Arbeitsanregungen zur Programmierung des Photowiderstandes

Aufgabe 4) In der Informationsdatei namens **ST_I_08_Arduino_Verzweigungen** wird der Befehl **`calibratePhotoResistor()`** vorgestellt. Mit diesem Befehl soll vor der Verwendung des Photowiderstandes in einer Programmschleife gemessen werden, zwischen welchen Werten die am analogen Steckkontakt A0 anliegende Spannung schwanken kann, je nachdem, ob der Photowiderstand beleuchtet oder abgedunkelt wird – dieser Messvorgang, bei dem der mögliche Wertebereich bestimmt wird, wird auch Kalibrieren genannt.

Der zugehörige Programmcode sieht so aus:

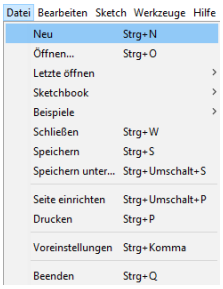
```
int sensorValue;
int sensorLow = 1023;
int sensorHigh = 0;

void calibratePhotoResistor(){
  sensorValue = analogRead(A0);
  if(sensorValue > sensorHigh){
    sensorHigh = sensorValue;
  }
  if(sensorValue < sensorLow){
    sensorLow = sensorValue;
  }
}
```

Analysiere dieses Unterprogramm, indem du die nachfolgende Tabelle ausfüllst und formuliere dann mit eigenen Worten, weshalb nach dem Kalibrieren in der Variablen **`sensorHigh`** der höchste (Spannungs-) Wert bzw. in der Variablen **`sensorLow`** der niedrigste (Spannungs-) Wert gespeichert ist, der am analogen Steckkontakt A0 angelegen ist:

Wert…	an Pin A0	von sensorValue	von sensorLow	von sensorHigh
…zu Beginn:	–	–	1023	0
nach dem…				
1. Schleifendurchlauf	850			
2. Schleifendurchlauf	720			
3. Schleifendurchlauf	912			
4. Schleifendurchlauf	400			
5. Schleifendurchlauf	356			
6. Schleifendurchlauf	187			
7. Schleifendurchlauf	567			
8. Schleifendurchlauf	991			
9. Schleifendurchlauf	985			
10. Schleifendurchlauf	412			
11. Schleifendurchlauf	105			

Aufgabe 5) An vielen Verkehrskreuzungen sinkt das Verkehrsaufkommen in der Nacht deutlich ab, sodass die Ampel von den gewohnten Phasen „grün“ – „grün blinkend“ – „gelb“ – „rot“ – „gelb“ – „grün“ – … auf „gelb blinkend“ umschaltet. Sobald es aber wieder hinreichend hell ist, zeigt die Ampel wieder die gewohnten Phasen.



Dies lässt sich an unserer „Ampelschaltung“ mit Hilfe des Photowiderstandes nachahmen – einige Teile des dazu benötigten Arduino-Programms findest du als „Codeschnipsel“ in der Informationsdatei **ST_I_08_Arduino_Verzweigungen**.

Erzeuge über den Menüpunkt `Datei – Neu` ein neues Arduino-Projekt, speichere es unter einem aussagekräftigen Namen (z.B. `Ampel_Nachtschaltung`) und ergänze das folgende Programmgerüst, sodass das vollständig Programm die Ampelschaltung so wie oben beschrieben steuert:

```
int redLedPin = 10;
int yellowLedPin = 5;
int greenLedPin = 3;
int sensorValue;
int sensorLow = 1023;
int sensorHigh = 0;

void calibratePhotoResistor(){
  ...
}

void ledBlink(int pin, int times){
  ...
}

void setup() {
  ...
}

void loop() {
  sensorValue = analogRead(A0);
  if(sensorValue > ...){
    ...
  }
  else{
    ...
  }
}
```

Teste dein Programm!

Aufgabe 6) Die nahegelegte Lösung zur Steuerung einer Ampel-Nachtschaltung aus der letzten Aufgabe berücksichtigt nur die Ampel für den fließenden Verkehr, nicht jedoch die Fußgängerampel, um die die Ampelschaltung in Aufgabe 2 ergänzt wurde. Überlege selbst, in welchen „Modus“ die Fußgängerampel in der Nachtschaltung wechseln könnte, um Fußgängern zu signalisieren, dass auch nachts beim Überqueren der Fahrbahn achtzugeben ist. Codiere die von dir erdachte Lösung und teste das Programm.