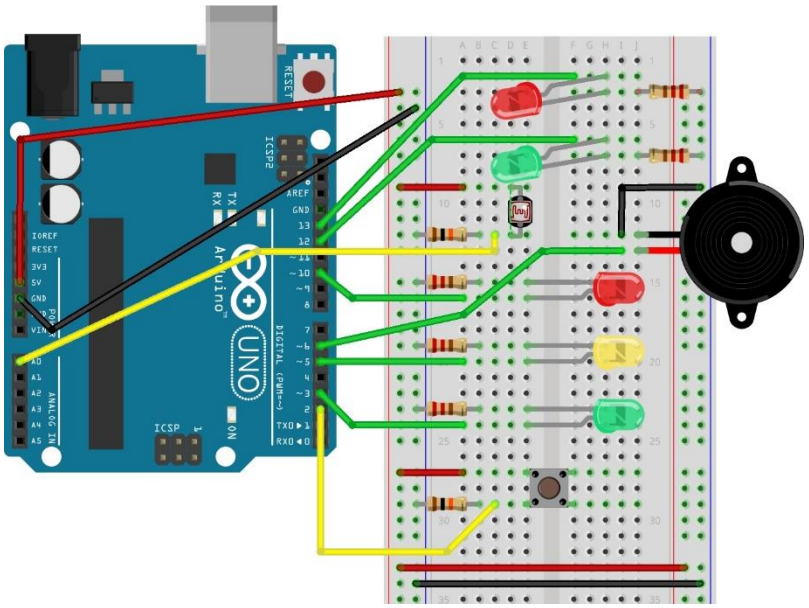
Arduino – Programmierpraxis: Messwerterfassung

Arbeitsanregungen zum Erfassen und Auswerten von Sensor-Messwerten

In den folgenden Arbeitsanregungen experimentierst du mit einigen Sensoren und den Daten, die von diesen Sensoren gemessen werden. Auch für diese Aufgaben kann zweckmäßigerweise wieder die „Ampelschaltung“ (vgl. Abbildung) weiterverwendet werden.



Aufgabe 1) In der Informationsdatei **ST_I_11Arduino_Praxis_Messwerterfassung** ist auf Seite 1 angegeben, wie Daten, die von einem Photowiderstand gemessen werden, im `loop`-Block eines Arduino-Programms in einer `int`-Feldvariablen gespeichert werden können.

a) Codiere in Anlehnung an den in der Informationsdatei abgebildeten Beispielcode ein Programm, das 100 Messwerte vom LDR-Widerstand (z.B. der „Ampelschaltung“) in einer int-Feldvariablen speichert und diese Messwerte entsprechend nebenstehender Abbildung in zehn Zeilen zu jeweils zehn Spalten am seriellen Monitor ausgibt.

```
271 260 260 258 276 649 751 765 775 775
749 599 401 336 315 296 308 306 456 795
825 833 833 834 834 835 834 835 834 834
834 833 828 825 807 628 592 530 347 318
309 300 312 298 301 298 310 356 768 819
824 828 827 827 827 827 824 812 809 826
827 828 829 829 829 830 831 832 832 835
833 835 833 834 835 835 835 836 837 836
837 836 836 838 837 837 837 838 838 836
836 835 838 835 835 834 833 834 832 831
```

Hinweis: Verwende dabei auch den Befehl **`calibratePhotoResistor()`** aus den **Aufgaben 4)** bis **6)** der Datei **ST_AA_08Arduino_Verzweigungen**.

b) Erweitere das Programm aus Aufgabe a) um die Frage-Programmblöcke zur Berechnung der Summe bzw. des arithmetischen Mittelwerts aus deiner Lösung zu Aufgabe 2) der Datei **ST_AA_10Arduino_Praxis_Feldvariable**. Teste diese Programmblöcke und gib die errechneten Werte am seriellen Monitor aus.

Zusatz: Vermutlich hast du die Lösungen zu den Aufgabenteilen a) und b) so codiert, dass ständig neue Messwerte gespeichert und angezeigt werden? Das ständige Aktualisieren der Anzeige am seriellen Monitor erschwert aber das Ablesen der Werte, oder?
Verändere den `loop`-Block so, dass neu gespeicherte Messwerte erst nach dem Drücken des Druckschalters am seriellen Monitor ausgegeben werden.

…du kannst das Programm aber z.B. noch so erweitern, dass je nach Größe des vom Photowiderstand gemessenen Wertes eine andere der drei Leuchtdioden (grün – gelb – rot) aufleuchtet...

c) Bei Messwerten interessiert man sich meist auch für den kleinsten bzw. den größten der gemessenen Werte. Wenn Messwerte in einer Feldvariablen gespeichert sind, kann die Berechnung (z.B.) des kleinsten Messwertes folgendermaßen erfolgen:
   – Zuerst wird in einer Variablen passenden Datentyps jener Wert als aktuell kleinster Wert gemerkt, der auf der Indexposition 0 der Feldvariablen gespeichert ist.
   – Sodann wird für jeden Wert der Feldvariablen ab Indexposition 1 überprüft, ob dieser Wert kleiner ist als der aktuell als kleinster Wert gespeicherte Wert. Wenn dies der Fall ist, wird der Wert an der jeweiligen Indexposition als neuer aktuell kleinster Wert gemerkt.

   Übersetze diese Berechnungsbeschreibung in einen Frage-Programmblock, der den kleinsten Wert in einer `int`-Feldvariablen berechnet, teste diesen Programmblock am Feld der Messwerte vom Photowiderstand und gib den kleinsten Messwert am seriellen Monitor aus.

d) Verändere die Berechnungsidee aus Aufgabenteil **c)** so, dass nicht der kleinste, sondern der größte Wert in einer **`int`**-Feldvariablen berechnet wird, erweitere dein Programm um den entsprechenden Frage-Programmblock und teste diesen...

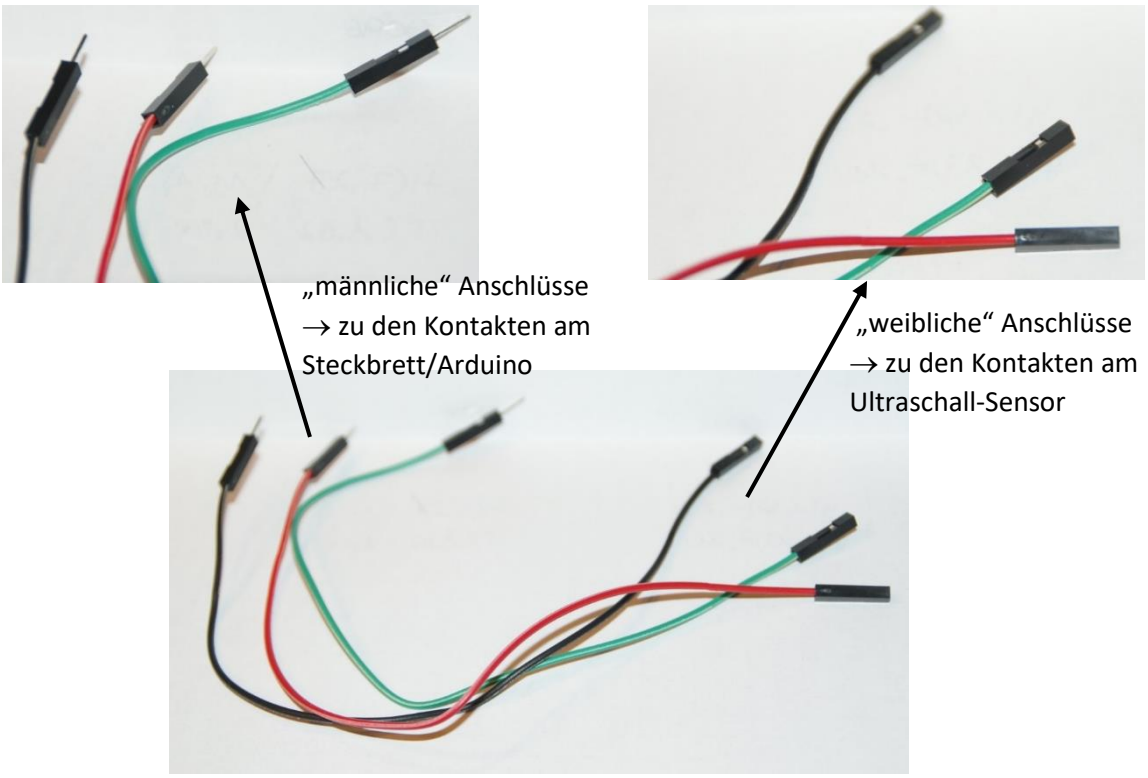
Aufgabe 2) Das Arduino-Programm, das du als Lösung zu **Aufgabe 1)** codiert hast, sollte leicht für die Messung, (kurzfristige) Speicherung und Auswertung von Daten anderer Sensoren angepasst werden können. Wenn du der nebenstehend nochmals abgebildeten Anregung aus der Informationsdatei gefolgt bist, ist im einfachsten Fall nur der jeweils andere Sensor am angegebenen analogen Port (im Beispielcode: A0) anzuschließen. Allenfalls ist aber höchstens die gelb unterlegte Codezeile durch den für den verwendeten Sensor passenden Code zu ersetzen. Beide Varianten sollen bei den folgenden Teilaufgaben ausprobiert werden

```
if(index < 5){
  intArray[index] = analogRead(A0);
  index = index + 1;
}
```

a) In der Informationsdatei **ST_I_11Arduino_Praxis_Messwerterfassung** werden die Verwendung und die Befehle für die Programmierung des **TMP36 Temperatursensors** erklärt. Ergänze deine Schaltung um diesen Sensor und verändere dein Programm zur Lösung von Aufgabe 4) so, dass die vom Temperatursensor gemessenen Daten zuerst in einer Feldvariablen gespeichert und diese Daten dann durch Berechnung der statistischen Maßzahlen „arithmetischen Mittel“ sowie „Minimum“ (kleinster Wert) und „Maximum“ (größter Wert) „analysiert“ werden. Miss mit diesem Sensor sodann die Temperatur deiner Finger, z.B. indem du den Sensor mit Daumen und Zeigefinger vorsichtig „drückst“.

b) In der Informationsdatei **ST_I_11Arduino_Praxis_Messwerterfassung** wird auch der **HC-SR04-Ultraschallsensor** zum Messen von Entfernungen vorgestellt. Damit die Entfernungsmessung einigermaßen flexibel erfolgen kann, sollte dieser Sensor mit Kabeln, die auf einer Seite einen

„weiblichen“ und auf der anderen Seite einen „männlichen“ Anschluss aufweisen, mit der Steckplatine verbunden werden – vgl. folgende Abbildungen:



Ergänze deine Schaltung aus Aufgabenteil a) um den HC-SR04-Ultraschallsensor und verändere dein Programm so, dass nun die über den Ultraschallsensor gemessenen Distanzwerte in einer Feldvariablen gespeichert und dann deren Mittelwert bzw. Minimum und Maximum berechnet und am seriellen Monitor ausgegeben werden.

Hinweis: Die Distanzmessung wird umso genauer, je größer das Objekt ist, zu dem der Abstand gemessen werden soll!

c) Die Schallgeschwindigkeit ist (unter anderem) von der Lufttemperatur abhängig. Wenn die Temperatur T in Grad Celsius gemessen wird, gilt in guter Näherung für die Schallgeschwindigkeit *v* (in Meter pro Sekunde):

$$v = 331{,}5 + 0.6 * T.$$

Kombiniere nun in einem neu codierten Programm die Temperaturmessung mit dem TMP36-Temperatursensor mit der Distanzmessung über den HC-SR04-Ultraschallsensor. Vergleiche dann die Abstandswerte, die mit der konstanten Schallgeschwindigkeit von 343,2 Meter pro Sekunde errechnet werden, mit jenen, die die Temperaturabhängigkeit der Schallgeschwindigkeit berücksichtigen, z.B. indem du die Werte in zwei Feldvariablen speicherst, diese dann am seriellen Monitor ausgeben lässt und so vergleichen kannst.

Arbeitsanregungen zum Codieren und Verwenden von Programmbibliotheken

Vermutlich hast du die letzten Aufgaben so gelöst, dass du die Programmblöcke für die Berechnung der statistischen Maßzahlen immer in die neuen Programmierprojekte kopiert hast oder ein älteres Programmierprojekt, in dem diese Programmblöcke schon codiert waren, unter neuem Namen gespeichert und dann die entsprechenden Programmteile für den jeweiligen neuen Sensor neu codiert hast.

In der Programmiersprache C können Programmblöcke, d.h. selbst codierte Befehle oder Fragen, aber auch in sogenannten Programmbibliotheken (engl.: Libraries) gesammelt werden. Wird dann eine solche Programmbibliothek in ein Programmierprojekt importiert, stehen alle dort codierten Befehle und Fragen genauso zur Verfügung, wie die Arduino-Standardbefehle.

Aufgabe 3) Mit einer Folienpräsentation und/oder in der Informationsdatei **ST_I_11Arduino_Praxis_ Messwerterfassung** wird genau erklärt, wie du eine Programmbibliothek namens **morse.cpp** mit den Programmblöcken für das Morsen aus Aufgabe 6) aus der Datei **ST_AA_06Arduino_Modularisierung** codieren und diese in ein Programmierprojekt einbinden kannst. Ein Programmgerüst für diese Programmbibliothek namens **morse_skeleton.cpp** und die fertig codierte zugehörige Header-Datei namens **morse.h** werden zur Verfügung gestellt.

a) Studiere die Erklärungen in der Informationsdatei,
   - erweitere das Programmgerüst so zur Programmbibliothek **morse.cpp**, dass sie der in der Informationsdatei **ST_I_11Arduino_Praxis_ Messwerterfassung** vorgestellten Version entspricht, und
   - teste deine Programmbibliothek, indem du sie wie in der Informationsdatei beschrieben in ein Programmierprojekt einbindest und die in der Bibliothek codierten Befehle und Fragen testest.

b) Erweitere nun die Programmbibliothek morse.cpp und die zugehörige Header-Datei so, dass Befehle für die Morsecodes ALLER Zeichen des Alphabets zur Verfügung stehen, und teste diese erweiterte Programmbibliothek mit Hilfe eines geeigneten Programmierprojekts.

Aufgabe 4) Für die Codierung einer Programmbibliothek, die die in den letzten Arbeitsanregungen verwendeten Frage-Programmblöcke zur Berechnung statistischer Maßzahlen beinhaltet, findest du das Codegerüst namens **iArrStatistics_skeleton.cpp** samt zugehöriger Header-Datei **iArrStatistics.h** im passenden Unterordner des Dateiordners für Arduino-Programmbibliotheken.

Vervollständige das Codegerüst der Programmbibliothek, indem du den Code deiner Lösungen zur **Aufgabe 1)** passend in das Codegerüst kopierst und die Datei unter dem Namen iArrStatistics.cpp speicherst. Teste diese Programmbibliothek, indem du sie in ein geeignetes Programmierprojekt einbindest.