Analoge Eingänge und die serielle Konsole

Bemerkung: Der Arduino muss bei den folgenden Programmen mit dem Computer verbunden bleiben. Alternativ kann später vielleicht ein LCD-Display eingesetzt werden.

Vorbereitung: Wie funktioniert ein Potentiometer?

Ein Potentiometer ist ein elektrisches Bauteil, das prinzipielle stets folgendermaßen aufgebaut ist: Zwei der für gewöhnlich drei Kontakte des Potentiometers sind durch einen Drahtwiderstand verbunden. Der dritte Kontakt schleift beim Drehen oder Schieben des Potentionemters über diesen Drahtwiderstand:

Das Schaltzeichen eines Potentiometers sieht entsprechend wie folgt aus, wobei der Schleifkontakt durch den Pfeil symbolisiert wird.

Werden die beiden Anschlüsse des Drahtwiderstands nun mit einer Spannung verbunden, kann man durch drehen des Schleifkontakts den Gesamtwiderstand in beliebige Teile „aufsplitten“. Entsprechende er Widerstände teilt sich dann auch die Spannung auf.

Beispiel: Angenommen, das Potentiometer hat einen Gesamtwiderstand von 2kOhm und die Anschlüsse 1 und 2 sind an einer Spannung von 5V angeschlossen. Der Schleifkontakt steht genau in der Mitte.

In diesem Fall halbiert der Schleifkontakt den Gesamtwiderstand und damit auch die Gesamtspannung. Man kann das Potentiometer im Ersatzsschaltbild wie im Bild zu sehen durch zwei Widerstände ersetzen und sich die Zusammenhänge auf diese Weise klar machen.

Aufgaben

Für das Protokoll...

- Überlege dir, wie die Werte der Teilwiderstände und der Spannungen sich verändern, wenn du zwei andere Stellungen des Schleifkontakts wählst. Zeichne jeweils ein Ersatzschaltbild.
- Miss den Gesamtwiderstand des Potentiometers
- Beobachte mit deinem Multimeter, wie sich der Widerstand zwischen dem Schleifkontakt und einem der Anschlüsse verändert, wenn du die Achse des Potis drehst.

Sketch 4.1

- Baue die Schaltung wie in der Abbildung auf
- Öffne den Sketch A041_Poti[1)]
- Starte das Programm.
- Öffne den „seriellen Monitor“ (STRG+UMSCHALT+M).
- Verändere die Stellung des Potentiometers und beobachte die Werte im Fenster des seriellen Monitors.

Neue Befehle

- `analogRead(pin)`: Liest den Wert eines festgelegten analogen Pins aus. Die resultierenden Integer Werte haben ein Spektrum von 0 bis 1023. (Bemerkung: Analoge Pins müssen im Gegensatz zu digitalen nicht zuerst als Eingang oder Ausgang deklariert werden.)
- `Serial.begin(9600)`:Öffnet den seriellen Port und setzt die Datenrate auf 9600 bps.
- `Serial.print` / `Serial.println`: Sendet eine Nachricht oder einen Wert an den seriellen Monitor.

```
Serial.print("Das ist eine Nachricht ohne Zeilenumbruch");''
Serial.println("Das ist eine Nachricht mit Zeilenumbruch am Ende");
Serial.print(sensorValue);  Sendet den Wert der Variable sensorValue.
```

Sketch 4.2

Rechne die ausgelesenen Werte so um, dass sie den Werten der am analogen Eingang anliegenden Spannung entsprechen. Ändere den Sketch so ab, dass auf dem seriellen Monitor der Spannungswert ausgegeben wird, wie z.B.: „Die Spannung beträgt: 2,1 V“ Speichere den Sketch unter dem Namen „A042_Poti_Spannung_2“ ab.

Beachte die Infos zu den Rechenoperationen. Experimentiere mit den Operationen und unterschiedlichen Variablentypen. Was passiert, wenn eine Wert den zulässigen Wertebereich eines Variablentyps verlässt? **Programmiere Beispiele, protokolliere deine Erkenntnisse!**

Info: Rechenoperationen

Informationen Rechnen mit dem Arduino

Arithmetische Operatoren umfassen Addition, Subtraktion, Multiplikation und Division. Sie geben die Summe, Differenz, das Produkt oder den Quotienten zweier Operatoren zurück.

```
y = y + 3;
x = x - 7;
```

```
i = j * 6;
r = r / 5;
```

Bemerkungen

- Die Operation wird unter Beibehaltung der Datentypen durchgeführt. `9/4` wird so zum Beispiel zu 2 und nicht `2,25`, da 9 und 4 Integer[2] sind und keine Nachkommastellen unterstützen.
- Dies bedeutet auch, dass die Operation „überlaufen“ kann wenn das Resultat größer ist als der Datentyp zulässt, dann kommen unsinnige Ergebnisse heraus.
- Wenn die Operanden unterschiedliche Datentypen haben wird der „größere“ Typ verwendet. Hat zum Beispiel eine der Zahlen (Operanden) den Datentyp 'float' und der andere 'int', so wir Fliesskomma Mathematik zur Berechnung verwendet.
- Wähle Variablentypen, die groß genug sind um die Werte der Ergebnisse zu speichern. Sei Dir bewusst an welcher Stelle die Werte überlaufen. Für Berechnungen die Brüche ergeben sollten immer 'float' Variablen genutzt werden. Allerdings mit dem Bewusstsein der Nachteile: Großer Speicherbedarf und langsame Geschwindigkeit der Berechnungen.
- Nutze den Form Operator z.B. `(int)myFloat` um einen Variablen Typen spontan in einen anderen zu verwandeln. Zum Beispiel wird mit `i = (int)3.6` die Variable `i` auf den Wert 3 gesetzt. So hast du die Kontrolle darüber, von welchem Typ das Ergebnis deiner Rechung letztlich ist.

Info: Gemischte Zuweisungen

Informationen Rechnen mit dem Arduino

Gemischte Zuweisungen kombinieren eine arithmetische Operation mit einer Variablen Zuweisung. Diese werden üblicherweise in Schleifen gefunden, die wir später noch genauer Beschreiben werden. Die gängigsten gemischten Zuweisungen umfassen:

```
x++          // identisch mit x = x + 1, oder Erhöhung von x um +1
x--          // identisch mit x = x - 1, oder Verminderung von x um -1
x+=y         // identisch mit x = x + y, oder Erhöhung von x um +y
x-=y         // identisch mit x = x - y, oder Verminderung von x um -y
x*=y          // identisch mit x = x * y, oder Multiplikation von x mit y
x/=y         // identisch mit x = x / y, oder Division von x mit y
```

Bemerkung: Zum Beispiel führt x *= 3 zur Verdreifachung des alten Wertes von 'x' und weist der Variablen 'x' des Ergebnis der Kalkulation zu.

1)

Herunterladen nach „sketches“, dann öffnen

2)

Ganzzahlen