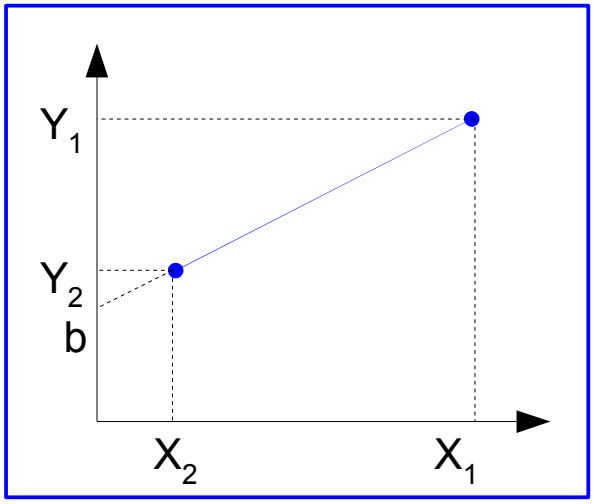
Analogwertberechnung mit stückweise linearen Funktionen

Bei der Analogwertverarbeitung können Sensorkennlinien oft nicht mit einer exakten Formel berechnet werden. Entweder ist der Polynomgrad für die Berechnung zu hoch oder eine Rechnung mit Fließkommazahlen zu langsam.

In diesen Fällen kann man den Ausgabewert mit stückweise linearen Funktionen einfach ermitteln. Nachteil ist hier die Ungenauigkeit zwischen den Stützstellen der Geraden. Durch geschickte Wahl der Stützstellen und des für die Anwendung benötigten Analogwertebereiches kann man trotzdem brauchbare Ergebnisse erzielen.

Gerade durch zwei bekannte Stützpunkte

Mit jeweils 2 bekannten Punkten (X_2, Y_2) und (X_1, Y_1) kann die Geradengleichung für diesen Abschnitt erstellt werden.



Die allgemeine Geradengleichung lautet bekanntlich

$$y = m * x + b$$

Die **Steigung m** ist der Quotient der Differenzen von Y und X. Die Differenz wird mit dem griechischen Buchstaben Δ (Delta) gekennzeichnet.

$$m = \frac{\Delta Y}{\Delta X} = \frac{Y_1 - Y_2}{X_1 - X_2}$$

Den **Y-Achsenabschnitt b** erhält man durch einsetzen eines der beiden Punkte in die Gleichung.

$$b = y_1 - m * x_1$$

Hinweis: wenn es sich bei der gesuchten Ergebnisgröße um einen physikalischen Wert (Spannung, Temperatur, Abstand) handelt, so sollte bei der Erstellung der Gleichung unbedingt die Einheiten eingesetzt werden. Damit erkennt man Fehler bei der Gleichungserstellung, wenn sich die Einheit z.B. als Bit^2 / Volt ergibt. Dann wurde bei der Bestimmung der Steigung der Zähler und Nenner vertauscht.

Umsetzung in einen C-Algorithmus:

Bei der Umsetzung in ein C-Programm (ohne die Verwendung von float-Variablen) muss besonders darauf geachtet werden, dass bei den erforderlichen Multiplikationen kein Bereichsüberlauf auftritt. Daher wird hier ein Type-Cast auf einen größeren Datentyp eingefügt.

Bei physikalischen Größen, die einen negativen Wert besitzen können (z.B. Temperatur, Spannung) muss ein Datentyp mit Vorzeichen (signed) verwendet werden.

Beispiel Temperaturmessung: aus der Sensorkennlinie und den Parametern der AD-Wandlung wurde mit einer Tabellenkalkulation eine Kennlinie erstellt. Je nach Anwendung wählt man den benötigten Temperaturbereich aus, z.B. 0-60°C. Für diesen Bereich soll jetzt die Kennlinie linearisiert werden. Hier ist der ADwert für 0°C = 48, für 60°C = 177.

Die gesuchte Formel soll als Ergebnis eine Temperatur in Abhängigkeit vom ADwert berechnen.

Die Steigung der Geraden ergibt sich damit zu

$$m = \frac{60\,°C - 0\,°C}{177\text{Bit} - 48\text{Bit}} = \frac{60\,°C}{129\text{Bit}}$$

Der Achsenabschnitt b ergibt sich zu

$$b = 60\,°C - \frac{60\,°C}{129\text{Bit}} * 177\text{Bit} = -22{,}32\,°C\ gerundet\ -22\,°C$$

Bei der Umsetzung im Programm werden die Einheiten natürlich weggelassen.

```
unsigned char ADwert;
signed int Temperatur;

     ADwert = ain(0);         // Messung Kanal 0
     // mit Type-Cast auf signed int, der Maximalwert der
     // Multiplikation ist 255*60 = 15300 und damit weit weg
     // vom Maximalwert +32767
     Temperatur = ( (signed int)ADwert * 60 ) / 129 – 22;
```

Mit der oben angegebenen Berechnung erhält man aufgrund der ganzzahligen Variablen die Temperatur mit einer Schrittweite von 1°C. Eine höhere Genauigkeit (etwa 0,45°C, begrenzt durch die Auflösung des AD-Wandlers) erhält man auch ohne float-Datentypen durch die Normierung auf 1/100°C oder 1/1000°C.

Bei 1/100°C entsprechen -22,32°C dem Wert -2232, bei der Multiplikation mit 6000 (entspricht 60,0°C) tritt jedoch ein Bereichsüberlauf bei int-Variablen auf => eine Zwischenvariable vom Typ **signed long** muss genutzt werden!

```
unsigned char ADwert;
signed int Temperatur;   // Endergebnis belegt 2 Byte
signed long Temp;        // Zwischenvariable, belegt 4 Byte

     ADwert = ain(0);         // Messung Kanal 0
     // mit Type-Cast auf signed long
     Temp = ( (signed long)ADwert * 6000 ) / 129 – 2232;
     Temperatur = (signed int) Temp; // verkürzen der Variablen
```

Der mögliche Wertebereich mit **signed int** reicht hier von -327,68°C bis 327,67°C und ist mehr als ausreichend.

Für die Ausgabe auf dem Display kann man Zahl in die Vor- und Nachkommastellen trennen und mit sprintf( ) einschließlich Komma wieder zusammenfügen. Die ganzen Berechnungen sind trotzdem um ein vielfaches schneller und platzsparender als die Verwendung von float-Variablen!

```
signed char Vorkomma, Nachkomma;
     Vorkomma = Temperatur/100;
     Nachkomma = Temperatur%100;  // Rest der Division
     sprintf(buffer,“Temperatur: %d,%02d“, Vorkomma, Nachkomma);
```