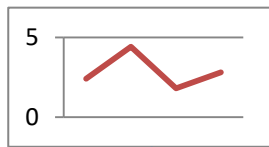


Analoge Eingänge

© technikum29



Der Eingang kann eine Spannung (Sensorwert) sein, die sich zwischen 0V und 5V ändert

Mit **analogRead(A0)** erhältst du eine Zahl, die beim 10-Bit A/D-Wandler zwischen 0 und 1023 liegt. Auf dem Seriellen Monitor wird die Zahl dezimal angezeigt.



Beispiel: Die Spannung an A0 beträgt 1,59V. Dann liefert **analogRead(A0)** den Wert 101000111 bzw. dezimal 327. Mit diesem Wert kannst du nun weiter arbeiten. Dazu sollte man ihm einen Namen geben, z.B.

```
int SensorWert = analogRead(A0)
```

PWM (Pulse Width Modulation):

Die Arduino-Ausgänge sind prinzipiell digital. Nur durch den Trick des schnellen Ein- und Ausschaltens können wir analoge Werte „simulieren“. Damit der Arduino weiß, wie lange die Ein- und Ausschaltedauer sein soll, muss man ihm das durch eine Zahl zwischen 0 und 255 mitteilen (8-Bit-Auflösung).

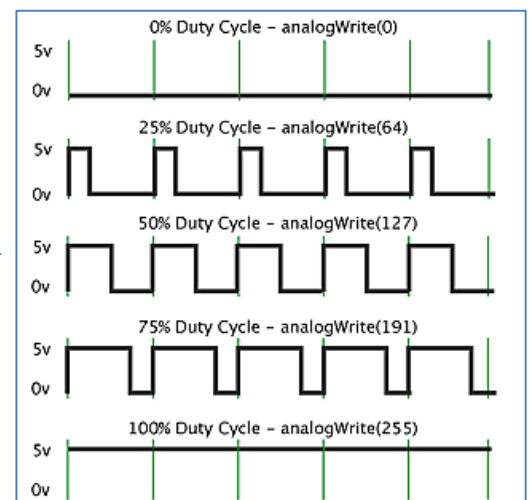
Dies geschieht durch **analogWrite(Pin-Nummer, Wert oder Variable)**.

Beispiel:

Durch **analogWrite(8, 127)** würde der Pin 8 die Hälfte der Zeit auf 0 stehen und die andere Hälfte auf 1. Eine angeschlossene LED würde ca. mit halber Helligkeit leuchten. Das Umschalten geschieht sehr schnell, siehe Zusatz von Aufg. 3, Batt 3.

Nicht alle Pins des Arduino sind PWM-fähig!

Zahlen (z) zwischen 0 und 255 mit **analogWrite(8, z)** eingeben



Duty Cycle = Arbeitszyklus