Lichtsensor

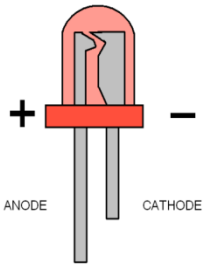
Als Photodetektor, auch Lichtsensor oder optischer Detektor, optoelektronischer Sensor, werden elektronische Bauelemente bezeichnet, die Licht unter Benutzung des photoelektrischen Effekts in ein elektrisches Signal umwandeln oder einen von der einfallenden Strahlung abhängigen elektrischen Widerstand zeigen. [https://de.wikipedia.org/wiki/Photodetektor]

Beliebte Lichtsensoren:

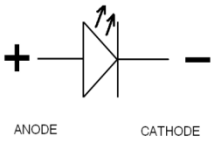
Photodioden:

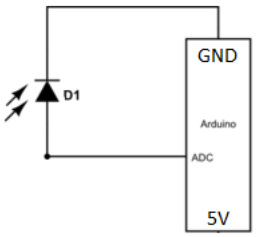
Photodioden erzeugen aus Licht einen Strom und eine Spannung. Die (Leerlauf) Spannung die sie erzeugen liegt bei “starkem Licht” im Bereich 200 bis 1200mVolt. Diese Spannung kann mit einem analogen Pin am Arduino gemessen werden.

Die Photodiode kann direkt am Arduino angeschlossen werden (Anode an AnalogInput und Kathode an Ground) oder mit einem Widerstand (zB. 1MOhm) gegen die Sperrrichtung betrieben werde.

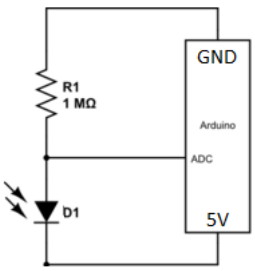


LED/Photodiode: kurzer Pin ist Kathode (minus), langem Pin ist Anode (plus)





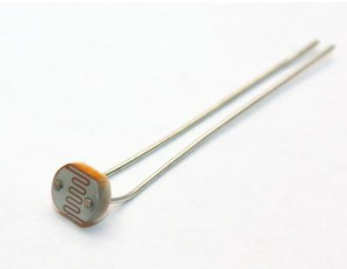
Direkt angeschlossen

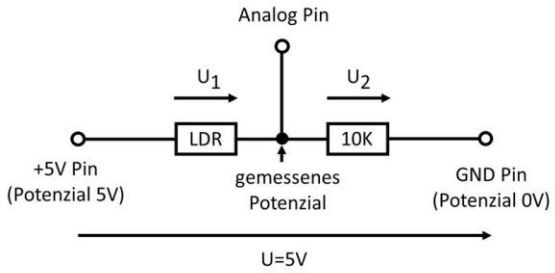


Gegen Sperrrichtung mit Widerstand

Auch eine normale (lichterzeugende) LED kann als Lichtsensor benutzt werden aber sie ist weniger als die Photodiode dafür geeignet.

Photowiderstand:





$$U_2 = U \cdot \frac{R_2}{R_1 + R_2}$$

Ein Photowiderstand (LDR Light Dependent Resistor) ändert abhängig vom einfallenden Licht seinen elektrischen Widerstand.

Je mehr Licht desto kleiner wird der Widerstand.

Z.B.: Der Widerstand des Photowiderstandes in der obigen Abbildung ist bei Dunkelheit bei 30kOhm und wenn es Hell ist bei 2kOhm.

Der Photowiderstand kann mit einem zweiten normalen Widerstand als Spannungsteiler benutzt werden um eine lichtabhängige Spannung zu erzeugen. Der Wert des zweiten Widerstandes muss passend zum Widerstandsbereich des Photowiderstandes gewählt werden.

Die Spannung im Spannungsteiler kann man anhand der Ohm'schen Gesetzte errechnen:

$U_2 = U_{ges} * R_2 / (R_1 + R_2)$

zB. Hell LDR = 2 kOhm:

Spannung U_2 = 5 Volt * 10 kOhm / (2 kOhm + 10 kOhm) = 4.16 Volt

zB. Dunkel LDR = 30 kOhm:

Spannung U_2 = 5 Volt * 10 kOhm / (30 kOhm + 10 kOhm) = 1.25 Volt

Übung:

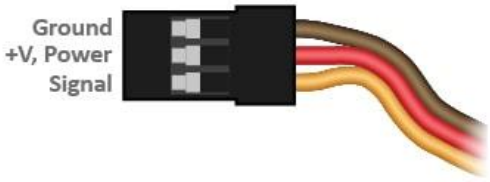
Geben sie die gemessenen Lichtsensorwerte mit Arduinos Serial Plotter aus.

Falls der mit analogRead(…); gelesene maximale Lichtsensorwert unter 200 liegt dann verändern sie in der Setup() Funktion mit analogReference(…) die ReferenzMessSpannung und Plotten die Werte erneut.

Modellbauservo

Ein Servo (lateinisch servus, „Diener, Sklave“) bezeichnet in der Elektrotechnik einen Verbund aus Ansteuerungs- und Antriebseinheit. Dies kann beispielsweise ein Elektromotor samt seiner Steuerelektronik sein. [https://de.wikipedia.org/wiki/Servo]

Servo



Servo-Connector

Servo-Horn

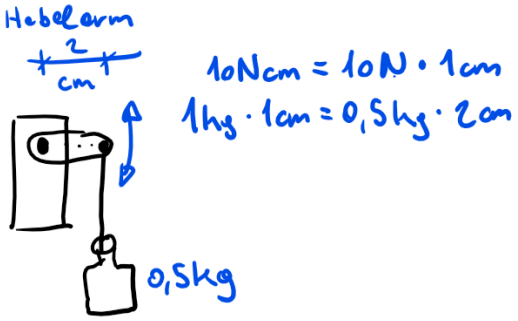
Ein Modellbauservo dreht sich normalerweise “nur” 180 Grad. Mit einem Pulsweiten Moduliertes Signal kann man den Servo von 0 bis 180 Grad positionieren. Praktischerweise gibt es dafür Arduino Libraries.

Es gibt aber auch Continuous-Rotation-Servos die permanent in dieselbe Richtung drehen können.

Für die Auswahl von Servos ist meist die benötigte Kraft und Größe ausschlaggebend.

Die Kraft des Servos ist sein Drehmoment und wird als Nm, Ncm oder kgcm angeben. (10N = ~1kg)

Sprich ein Servo mit 5Ncm kann 0,5kg bei einem Hebelarm von 1cm noch heben oder 0,05kg bei eine Hebelarm von 10cm. Ein Servo mit 75Ncm kann ein Gewicht von 7,5kg bei einem Hebelarm von 1cm noch heben oder 0,75kg bei einen Arm von 10cm.



Generell braucht ein Servo mehr Strom und ist grösser desto Stärker bzw. desto mehr Drehmoment er aufbringen kann.

Das Material des Getriebes im Servo ist mit ausschlaggebend wie stark ein Servo sein kann.

Kunststoff Getriebe ist Leiser.

Metall Getriebe kann mehr Kraft aufwenden.

Der Unterschied von analogen zu digitalen Servos ist (nur) wie sie intern den Motor für die Positionierung steuern. Digitale Servos sind meist etwas schneller und positionieren genauer. Beide Servo-arten werden aber auf die gleiche Weise (von außen) angesteuert.

Hinweis Stromverbrauch: Achten sie darauf das der Servo mit einer ausreichenden Spannungs- und Stromquelle versorgt ist.

Vermeiden sie es den Servo an dieselbe Stromversorgung wie den Arduino an zuschließen. Der vom Servo benötigte Strom kann an sonst analoge Messungen am Arduino stören.
(Achten sie darauf dass die Grounds bei separater Stromversorgung der Servos miteinander verbunden sind.)

Arduino Servo.h

Mit der Arduino Library Servo.h kann ein Modellbau-Servo einfach gesteuert werden.

Inkludiert wird die Library am Anfang ihres Programm-Codes mit dem Include-Statement.

```
#include <Servo.h>
```

Eine Variable bzw. Instanz eines Servos legen sie (global) so an:
`Servo myServo; // This is your servo instance named myServo`
myServo ist der Name ihrer Servo Variable.

Mit der **attach** Funktion setzten sie (in der `void setup()` Funktion) den Pin an dem der Servo am Arduino angeschlossen ist.
`myServo.attach(pin) // Attach myServo your pin number`

Es kann auch die Pulsweite des Servo für den Bereich 0 bis 180 Grad gesetzt werden.
`myServo.attach(pin, min, max) // Attach myServo your pin number and set the pulse width`

Mit der **write** Funktion steuern sie den Servo.
`myServo.write(45); // Servo go to position 45 (Degree)`

Mit der **writeMicroseconds** Funktion können sie den Servo höher aufgelöst positionieren.
`myServo. writeMicroseconds(1230); // Servo go to position of pulse width 1230`

https://www.arduino.cc/en/Reference/Servo

Übung:

Schreiben sie ein Programm das einen Servo hin und her bewegt.

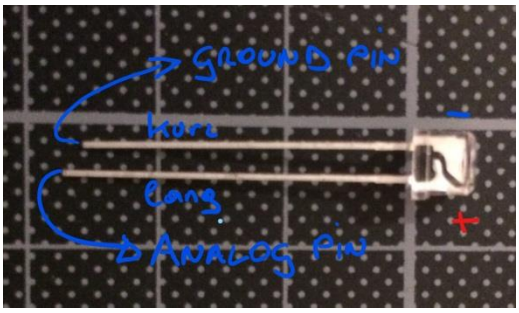
1D Light Scan Robot

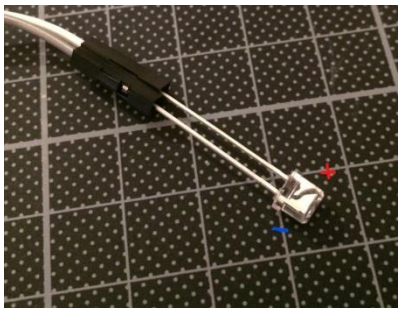
Was ist der 1D Light Scan Robot ?

Der 1D Light Scan Roboter scannt die Lichtstärke in seinem (Arbeits/Freiheits-) Bereich und verweilt etwas Zeit wo er die höchste Lichtstärke detektier hat. Er besteht aus einem Servo und einem einzigen Lichtsensor und wird von einem Mikrokontroller/Arduino gesteuert.

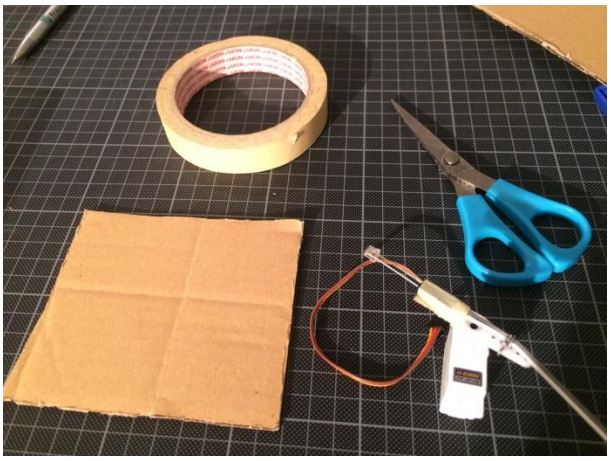
Bauanleitung

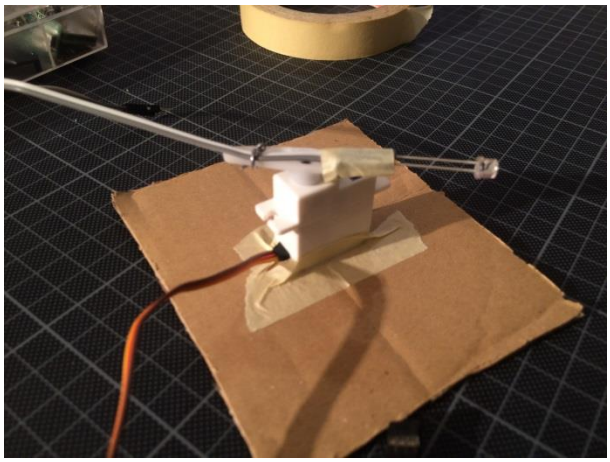
Benötigt wird ein Lichtsensor (bevorzugt eine Photodiode oder Photozelle), ein paar Jumper-Kabel (Female to Male und Male to Male), ein Modellbauservo, etwas Klebeband (Malerklebeband) sowie ein Stück Karton für die Bodenplatte. Sie sollten den Servo mit einem separaten Netzteil mit Strom versorgen.





1.) Stecken sie 2 Female to Male Kabel an den Lichtsensor.
2.) Befestigen sie den Lichtsensor mit etwas Klebeband (oder Faeden) am Servohorn so das er in der Drehenbene nach aussen zeigt.





3.) Befestigen sie den Servo mit etwas Klebeband an der Bodenplatte.
4.) Stecken sie das Servo-Horn mit Lichtsensor auf den Servo.

5.) Schließen sie die Steuer und Sensor Kable an ihren Arduino an.
   - Sensor-minus (Ground) an einen Ground Pin.
   - Sensor-plus an einen Analog-Pin (zB: A0)
   - Servo-Steuersignal (gelbes) (mit einem Male to Male-) Kabel an einen Digital-Pin (zB: 5)
6.) Schließen sie Strom/Power Leitungen (mit Male to Male Kabel) an ihre (5Volt) Strom/Spannungsversorgung
   - Braunes/Schwarzes Kabel an die minus (Ground) Leitung der Versorgung
   - Rotes Kabel an die Plus Leitung
7.) Verbinden sie die Ground-Leitung der Servo-Stromversorgung an einen Ground Pin am Arduino

Gratuliere, somit sollte die Mechanik und Elektronik fertig sein!

Steuer-software

In den Begleitunterlagen Lichtsensor und Modellbau-Servo ist das lesen von analogen Signalen und das Steuern von Servos beschrieben.

Mit analogRead() lesen sie die Spannung die der Lichtsensor produziert.

Mit servo.write() steuern sie die Servo Position.

Inkludieren sie Servo.h, legen eine Servo Variable/Instance im globalen-scope an und „Attachen“ sie den Servo und initialisieren sie den analog Pin als INPUT in der `void setup()` Funktion.

```
#include <Servo.h>

Servo myServo;              // Your Servo variable/instance

void setup() {
       myServo.attach(5);   // Attach servo PWM signal to pin 5
       pinMode(A0, INPUT);  // Set analog Pin A0 as INPUT
       Serial.begin(9600);  // Additionally let initialize a Serial-Output
}

// If you connect your Sensor or Servo to different pins then adapt them in your code too
```

Um ein Gefühl für die Lichtsensorwert Charakteristik zubekommen sollten sie für den Anfang in der `void loop()` Funktion den Sensorwert einlesen und per Serial.println(…) an den Serial Plotter ausgeben.

Falls die Sensorwerte sehr niedrig sind und auch der maximale Wert kleiner ist als 200 dann können sie die analoge Mess-Referenzspannung mit `analogReference(INTERNAL);` in der `void setup()` Funktion verändern.

Übung 1: Direkte Steuerung durch den Sensorwert

Schreiben sie ein Programm das direkt den Lichtwert für die Servo-Positionssteuerung (0-180) verwendet.

Plotten sie auch ihren Sensorwert mittels Serial Plotter.

Eine hilfreiche Funktion um die gelesenen analog Wert auf eine Servo Position um zurechnen und zu skalieren ist die map(...) Funktion.

zB: Ihr Lichtsensor liefert Werte von minimal 55 bis maximal 143 dann können sie diese mit er

```
int servoMappedPos = map(sensor_val, 55, 143, 0, 180);// Map to  0 – 180
```

auf den Bereich von 0 bis 180 mappen.

Zusätzlich sollten sie den gemappten Wert beschränken.

```
if (servoMappedPos < 0) servoMappedPos = 0; // If smaller then 0
else if (servoMappedPos > 180) servoMappedPos = 180; // If bigger then 180
```

Ihre loop Funktion könnte (mit Pseudocode) so aussehen:

```
void loop() {

	// 1.) Lesen sie den Sensor und weise sie ihn einer Variable zu

	// 2.) Geben sie den Wert mit Serial.println(…) aus

	// 3.) Rechnen sie den Sensorwert mit der map Funktion auf den Bereich 0 – 180
	// mit der map(value, fromLow, fromHigh, toLow, toHigh) um
	// und weisen diesen Wert einer neuen Variable zu

	// 4.) Beschränken sie den neuen Wert sicherheitshalber auf den Bereich 0-180

	// 5.) Senden sie den Positionswert mit myServo.write(…) an den Servo

	delay(1); // 6.) und gönnen wir uns eine kleine Pause
}
```

Übung 2: Licht im Bereich 0-180 scannen

Schreiben sie eine Funktion die den Servo in 1 Grad-Schritten von 0 – 180 fahren lässt.

Lesen sie nach jedem 1Grad Schritt einen Lichtsensor wert und merken sie sich die Position und den Wert falls dieser ein maximaler Wert ist. Die Funktion sollte die Position mit dem höchsten Lichtwert zurückgeben (returnen).

Ihre Funktion könnte (mit Pseudocode) so aussehen:

```
int ScanMaxLightAndReturnMaxPos() {

	// 1.) Positionieren sie den Servo in der Ausgangsposition 0 und warten 500ms

	// 2*) Legen sie Variablen zum Merken der des maximalen wertes und der Position an
	int maxLightValue = 0;
	int positionWithMaxLightValue = 0;

	// 3*) Durch laufen sie eine Schleife  180-mal
	for (int i = 0; i < 180; i = i + 1)
	{

		// 4.) Senden sie die Position i an den Servo und warten sie 10ms

		// 5.) Lesen sie den Lichtwert und merken diesen in einer Variable val

		// 6*) Vergleichen sie ihren aktuellen maximal wert mit dem aktuellen
		// und merken sich Wert und Position falls dieser höher ist
		if (val > maxLightValue) // check if new light value is higher
		{
			maxLightValue = val;		// update our highscore
			positionWithMaxLightValue = i;	// store the position
		}

		// 7*) Geben sie den aktuellen Licht wert und position aus
		Serial.print(i);	// Output servo position
		Serial.print(" ");	// Delimit between values
		Serial.println(val); // Output light value
	}

	// 8*) Geben sie die Position mit dem Maximalwert zurück
	return positionWithMaxLightValue;
} // n.) Ergänzen sie den fehlenden Code
```

Ihre neue loop Funktion könnte dann so aussehen:

```
void loop() {
	int maxPos = ScanMaxLightAndReturnMaxPos(); // Get max light position

	myServo.write(maxPos); // Fahren wir zum Licht

	delay(3000); // und gönnen wir 3 Sekunden Pause
}
```

Optionale Übung 3: Verbessern sie das Sensorsignal

Oft hat man es mit nicht idealen Sensorwerten. Rauschen oder zu geringe Auflösung erschweren das Arbeiten mit den Werten. Um dies zu verbessern gibt es viele Möglichkeiten. (Filtern, Mehrfach-Samplen, Offset rausrechnen, etc...)

Wichtig dafür ist es sich die Charakteristik der Werte und des Sensor anzusehen um nicht am Ziel vorbei Zuschießen oder es gar zu verschlechtern.

Filtern:

Die Signal Vergangenheit mit einbeziehen um keine zu schnellen Sprünge zu machen

```
valueTmp = (valueTmp * 2 + val) / 3;
```

Mehrfachsamplen:

Die Summe aus mehrfachen Sensor Werten bilden und wieder dividieren.

```
for (int i = 0; i < oversampling; i++)
{
	int tmp = analogRead(pin);

	value += tmp;

}

value = value  / oversampling; // Der Summen Wert muss nicht durch die volle Anzahl
                               // Samples dividiert werden. Zb: (oversampling – 2)
```

Minimale und Maximale Samples verwerfen:

Sie können auch den kleinsten und größten gelesen Wert nicht in ihre oversampling Summe einbeziehen

```
int minVal = 10000;
int maxVal = -10000;
// Collect all values
for (int i = 0; i < oversampling; i++)
{
	int tmp = analogRead(pin);
	value += tmp;
	// store the min and max value
	if (tmp < minVal) minVal = tmp;
	if (tmp > maxVal) maxVal = tmp;
}

value -= minVal + manVal;

value = value  / (oversampling - 2); // Der Summen Wert muss nicht durch die volle Anzahl
                                     // Samples dividiert werden. Zb: (oversampling – 4)
```