Übungen mit Lichtsensor und LED

Vor Beginn der Arbeiten solltest du die ***Elektrotechnischen Grundlagen*** aus folgenden Dokumenten durchlesen und verstanden haben:

- **Reihen- und Parallelschaltung**
- **Spannungsteiler**

In diesem Projekt nutzen wir einen selbst gebauten ***Spannungsteiler*** aus einem gewöhnlichen Widerstand und einem ***LDR*** (***Light Dependent Resistor*** – Lichtabhängiger Widerstand).

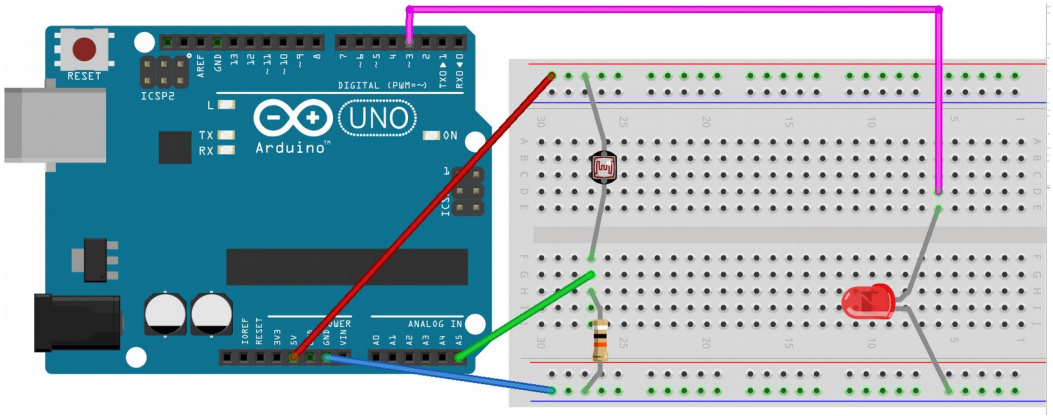
Projekt Ampel

Hardware

Du benötigst

1 LDR (Dunkelwiderstand > 1MΩ)
1 Widerstand (50-100kΩ)
4 Kabel (male/male)
1 Steckbrett
1 Arduino (Uno)

Aufbau



Stecke den LDR, den Widerstand und die LED wie im Bild auf das Steckbrett.
Verbinde den 5V-Pin des Arduino mir der oberen roten Querreihe. *(rotes Kabel im Bild)*
Verbinde einen GND-Pin des Arduino mit der unteren blauen Querreihe. *(blaues Kabel im Bild)*
Verbinde den analogen Eingangs-Pin A5 des Arduino mittig mit dem LDR und dem Widerstand. *(grünes Kabel im Bild)*

Bedenke für später, dass die analogen Eingangs-Pins des Arduino beim Programmieren anders heißen: Dort wird nach den digitalen Pins einfach weiter gezählt:

- letzter digitaler Pin: 13
- erster analoger Pin A0: 14
- letzter ***analoger Pin A5:*** ***19***

Grundlegende Funktion

1. Erzeuge in BlueJ interaktiv durch Rechts-Klick ein Objekt der Klasse ***PinAnalog*** an Pin-Nummer 19 *(=A5)*.
2. Rufe interaktiv die ***Methode*** `readValue()` auf und merke dir den erhaltenen Helligkeits-Wert.
   Halte nun eine Hand etwa 5cm über den LDR und wiederhole die Messung. Der neue Wert sollte kleiner sein, da du den LDR nun etwas abgedunkelt hast.
   Lege nun einen Finger ganz auf den LDR und wiederhole die Messung. Der Wert sollte nun nahe Hundert oder gar darunter liegen.
   Falls du ein Handy bei der Hand hast, so leuchte mit dessen Taschenlampe direkt auf den LDR und wiederhole die Messung. Der Wert sollte nun nahe Tausend oder gar darüber liegen.
3. „Zerstöre“ nun das erzeugte Objekt, indem du in BlueJ die virtuelle Maschine neu startest oder BlueJ schließt und neu startest.

Beleuchtung je nach Bedarf

Wir wollen nun ein Programm schreiben, das die Helligkeit einer LED je nach Bedarf einstellt:
Ist die Umgebung dunkel, so soll die LED hell leuchten und umgekehrt.

Dazu solltest du aber vorher schon einmal die entsprechende Übung zum Dimmen einer LED durchgemacht haben.

4. Schreibe eine ***Klasse LDR_dimmed_LED***. Darin brauchst du ein Referenz-Attribut vom Typ PinAnalog für den LRD und ein weiteres Referenz-Attribut vom Typ PinPWM für die LED.
5. Erzeuge die Objekte in deinem ***Konstruktor*** an Pin 19 (LDR) und Pin 3 (LED).
6. Schreibe eine ***Methode*** `dimm()`, die folgendes tut:
   Zuerst wird eine lokale ganzzahlige Variable `helligkeit` deklariert und darin der Sensor-Wert des LDR gespeichert … aber VORSICHT:
   Da dieser Wert von 0-1023 reicht, die prozentuale Helligkeit der LED aber nur von 0-100 regelbar ist, rechnen wir den Messwert entsprechend auf den Bereich 0-100 um:

```
int helligkeit = this.ldr.getValue();
helligkeit = (int)( helligkeit / 10.23 );
```

   Nun stellen wir die Helligkeit der LED auf den „Gegenwert“ der Umgebungs-Helligkeit ein:

```
this.led.setDutyCyclePercentage( 100 – helligkeit );
```

   Verständnis-Fragen:
   1) Wieso steht vor dem ersten „helligkeit“ ein int ?
   2) Wieso bekommen „led“ und „ldr“ ein this. vorne dran, „helligkeit“ aber nicht ?
   3) Was bewirkt wohl das „(int)“ vor „(helligkeit / 10.23)“ ?
7. Schreibe nun eine ***Methode*** `autoDimm()`, die folgendes tut:
   In einer gezählten Wiederholung soll 100 Mal folgendes ausgeführt werden:
   Erst soll die Methode `dimm()` aufgerufen werden.
   Gleich danach soll 300 Millisekunden gewartet werden `(Arduino.pause(300))`.
8. Erzeuge ein Objekt deiner Klasse und teste die Methode. *(Sie läuft jeweils 30 Sekunden lang.)*