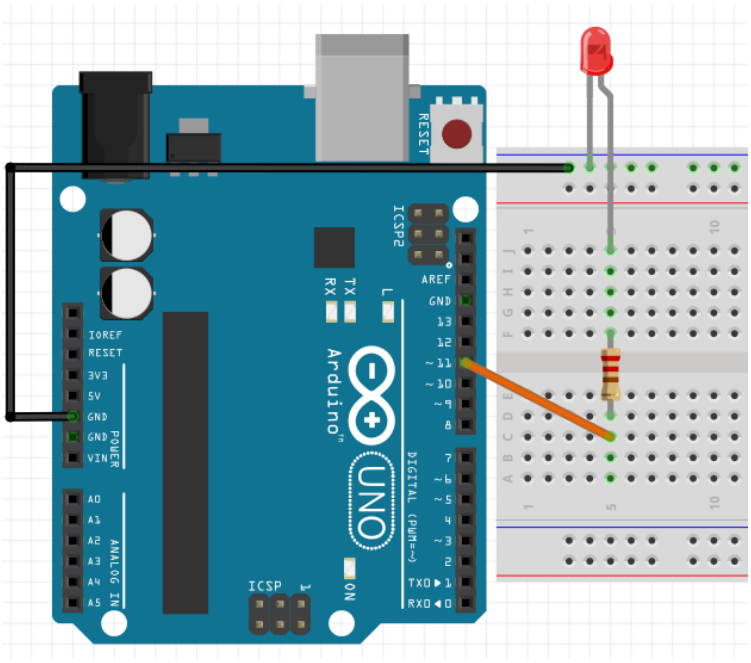
1 Blink



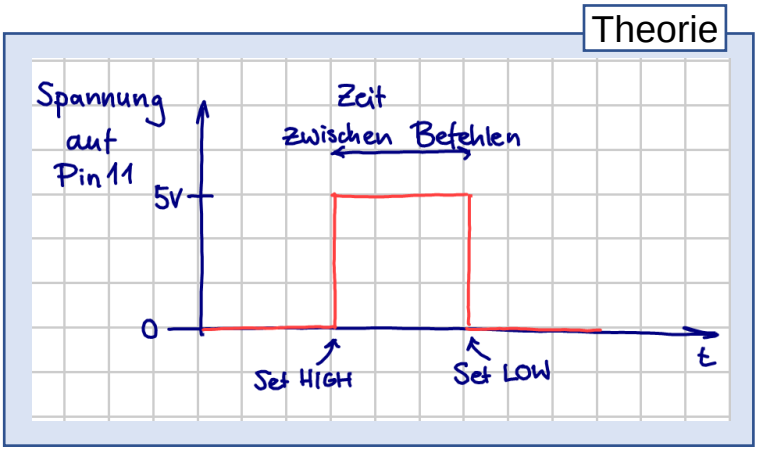
Beschreibung

Wir starten damit eine LED mit “HIGH” und “LOW” anzusteuern. Dadurch werden wir die LED an- oder ausschalten. Zusammen mit einer Pause (delay) zwischen den Schaltungen können wir die LED unterschiedlich schnell blinken lassen.

Komponenten

- 1x LED
- 1x Widerstand

Theorie



Neuer Code

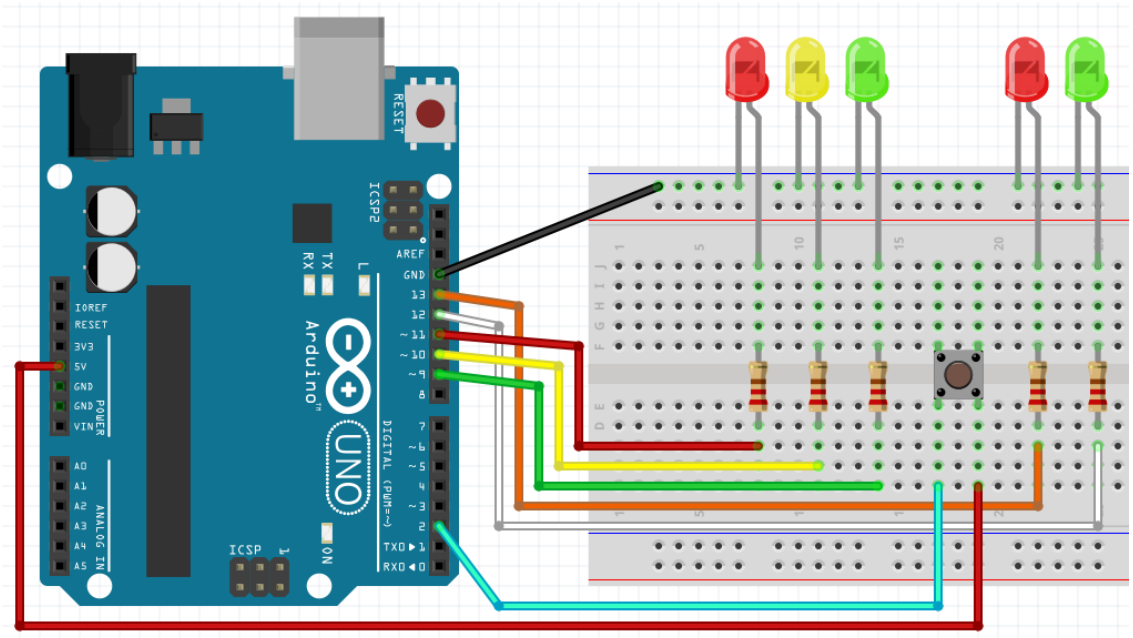
```
pinMode(11, OUTPUT);     // sets the digital pin 11 as output

digitalWrite(11, HIGH); // sets the digital pin 11 on

digitalWrite(11, LOW);  // sets the digital pin 11 off

delay(1000);            // waits for 1000 milliseconds = 1 sec
```

2 Traffic Light

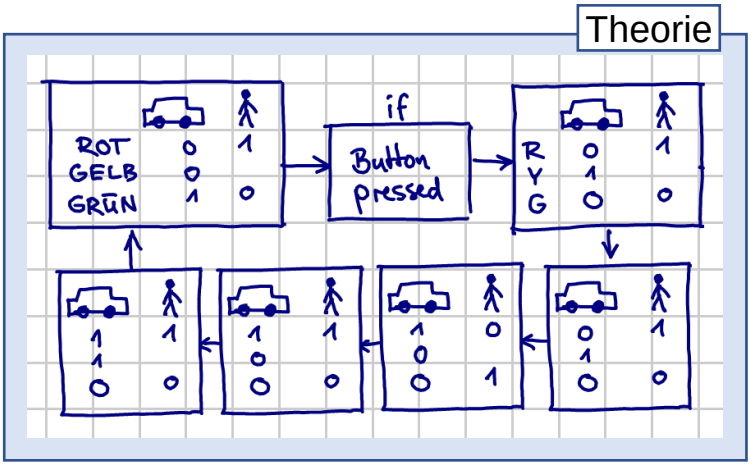


Beschreibung

Wir wollen anfangen **Logik** in unser System zu implementieren. Die Fußgängerampel soll grün werden, wenn wir den Knopf zum Überqueren der Straße drücken. Aus Sicherheitsgründen sollte die Ampel für die Autos vorher rot werden.

Komponenten

- 5x LED
- 5x Widerstand
- 1x Schalter



Neuer Code

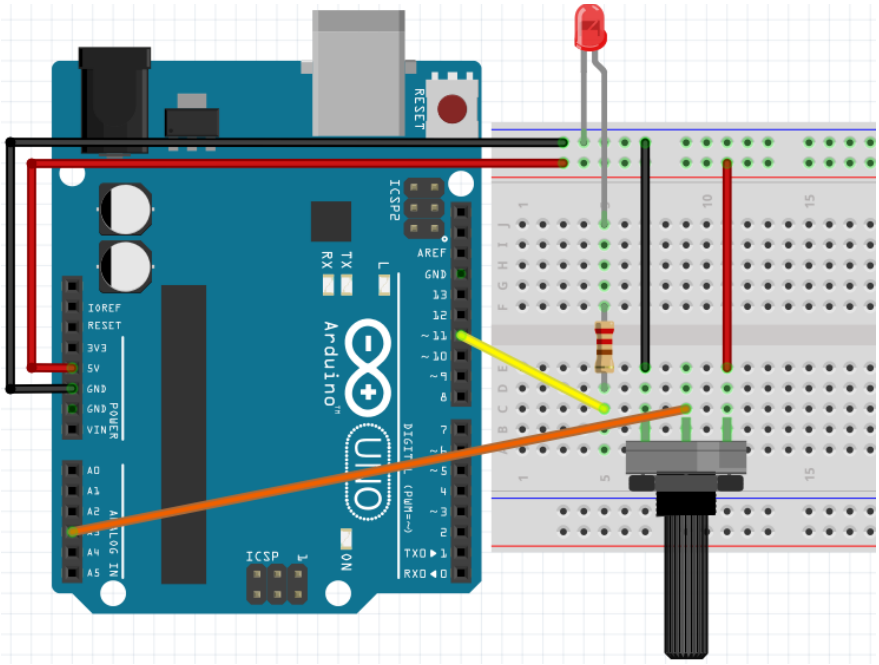
```
int x = 0;              // creates a variable that can hold a number

x = x + 5;              // adds 5 on x ; simple mathematical operations: + - * /

x = digitalRead(2)      // reads on Pin 2 and saves it into the variable
                        // digitalRead() returns either HIGH or LOW

if (x == y) {           // runs the following code only if x is equal to y
  // code to run        // other possible comparisons: bigger:(x > y) less:(x < y)
}                       // bigger or equal:(x >= y) unequal: (x != y)
```

3 Fade



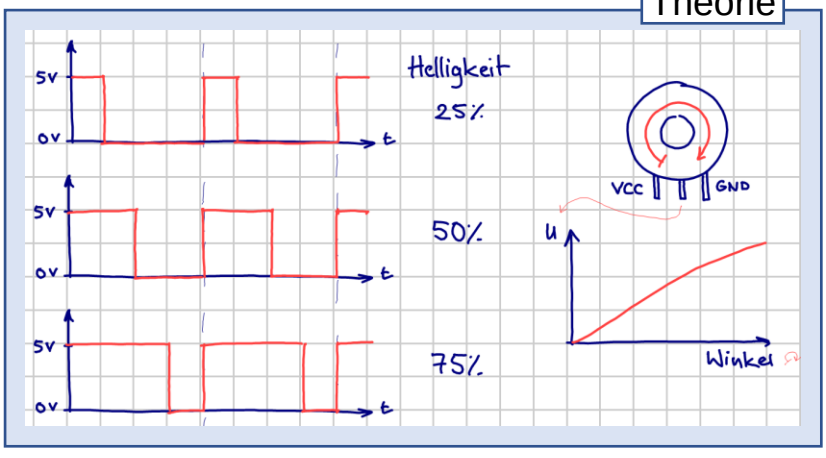
Beschreibung

Eine weitere Methode, unsere LED zu betreiben, heißt Pulse Wide Modulation oder kurz **PWM**. Dies gibt uns die Möglichkeit, die Helligkeit zu regulieren. Dies funktioniert durch sehr schnelles Ein- und Ausschalten der LED. Wenn wir die "on time" verlängern, erreichen wir eine höhere Helligkeit. Wir benutzen ein Potentiometer als Eingang.
Die Pins, die diesen Modus verwenden können, sind mit ~ gekennzeichnet.

Komponenten

- 1x LED
- 1x Widerstand
- 1x Potentiometer

Theorie



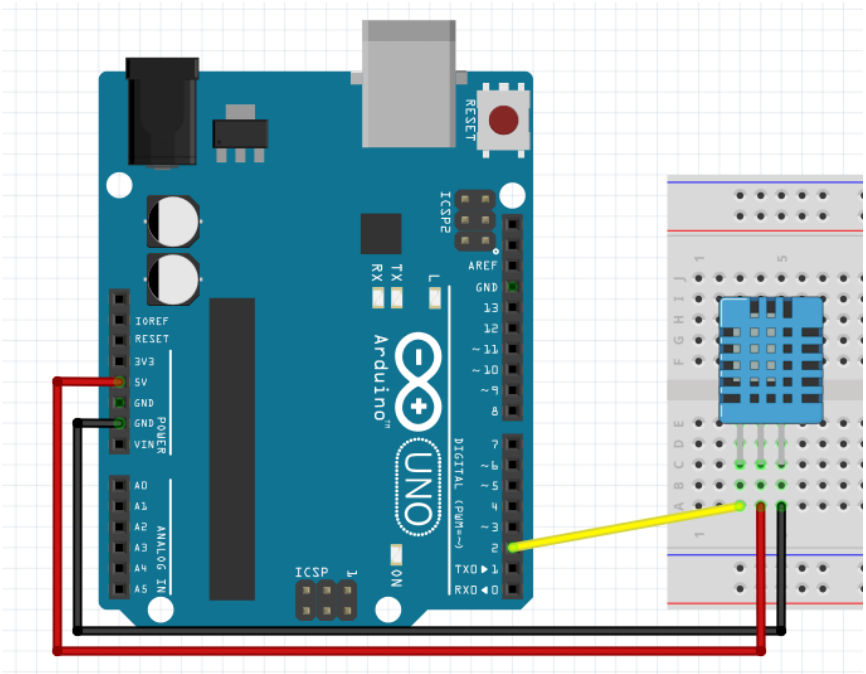
Neuer Code

```
val = analogRead(3);  // Reading the voltage on Analog Pin 3
                      // the value will be between 0 and 1023

bright = map(val, 0, 1023, 0, 255); // linear Transformation
                    // converting val from (0-1023) to (0-255)

analogWrite(11, bright); // writes the brightness from 0 to 255
```

4 Humidity Sensor



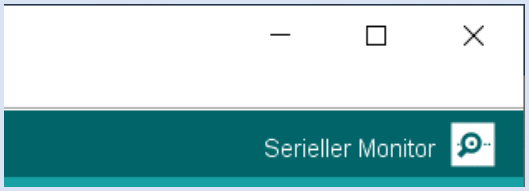
Beschreibung

Der nächste große Schritt ist die Verwendung komplexerer Sensoren in unserem System. Bei **digitalen Sensoren** können wir durch die Verwendung einer Bibliothek viel Zeit sparen. Sobald wir die richtige Bibliothek gefunden und installiert haben, ist es eine gute Idee, mit dem Beispielcode zu beginnen. Um die Werte zu sehen, die der Sensor erzeugt, senden wir die Daten über USB an unseren PC. Dies geschieht einfach über die serielle Kommunikation wie unten beschrieben.

Komponenten

- 1x Humidity Sensor DHT11

Tipp



Öffnen den SerialMonitor rechts oben auf der Arduino IDE um die Daten zu sehen

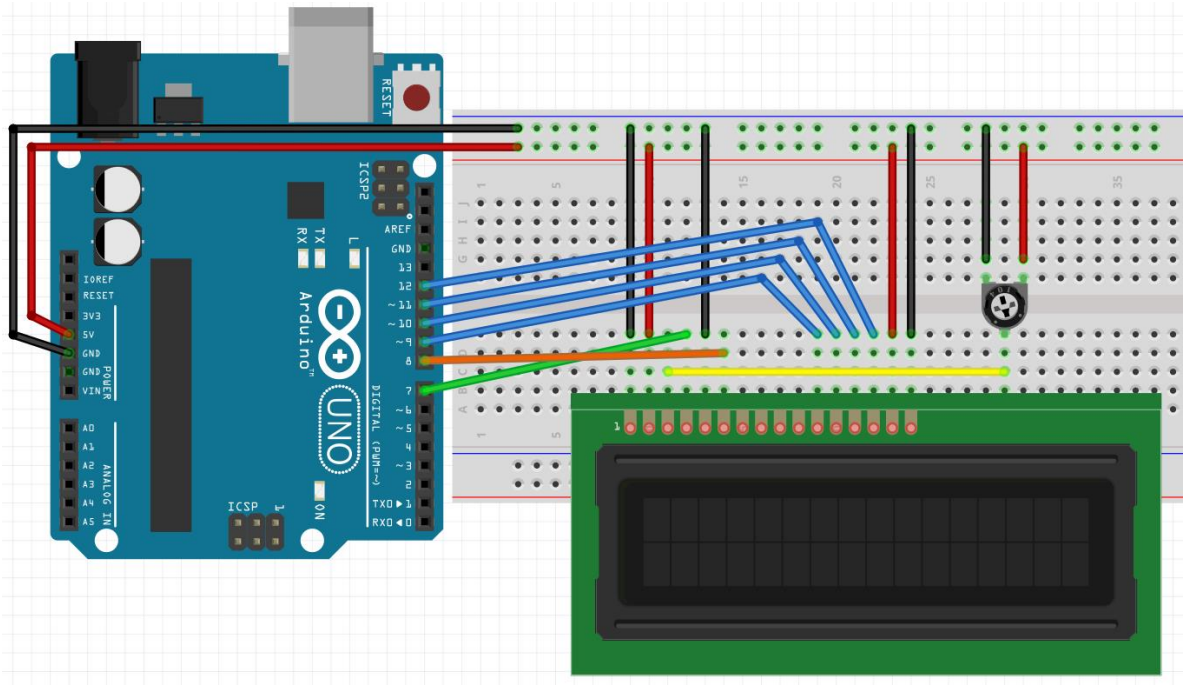
Neuer Code

```
#include <dht_nonblocking.h>   // includes a library in our scetch

Serial.begin( 9600);           // starts Serial Communication

Serial.print( "T = " );            // sends the characters T =
Serial.println( temperature, 1 ); // sends the value of temperature
                                  // and enters a new line in console
```

5 LCD Display



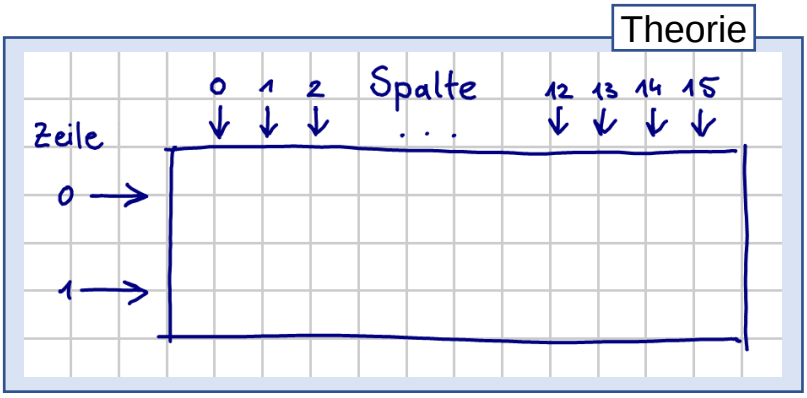
Beschreibung

Mit diesem **L**iquid **C**rystal **D**isplay können wir auf zwei Zeilen Zeichen ausgeben. Das Potentiometer dient hierbei zur Einstellung des Kontrasts.
Zur einfachen Bedienung verwenden wir die Bibliothek <LiquidCrystal.h>
Anwendung könnte ein solches Display darin finden Sensordaten live anzuzeigen, aber auch ein Mini-Spiel wäre denkbar.

Komponenten

- 1x LCD Anzeige
- 1x Dreh-Widerstand

Theorie



Neuer Code

```
LiquidCrystal lcd(7, 8, 9, 10, 11, 12); // initialize our LCD with the interface pins

lcd.begin(16, 2);                       // set up the LCD's number of columns and rows

lcd.print("Hello, World!");             // Print a message to the LCD.

lcd.setCursor(0, 1);                    // set the cursor to column 0, line 1

lcd.print(millis() / 1000);             // print the number of seconds since reset
```

6 Project: Solartracker

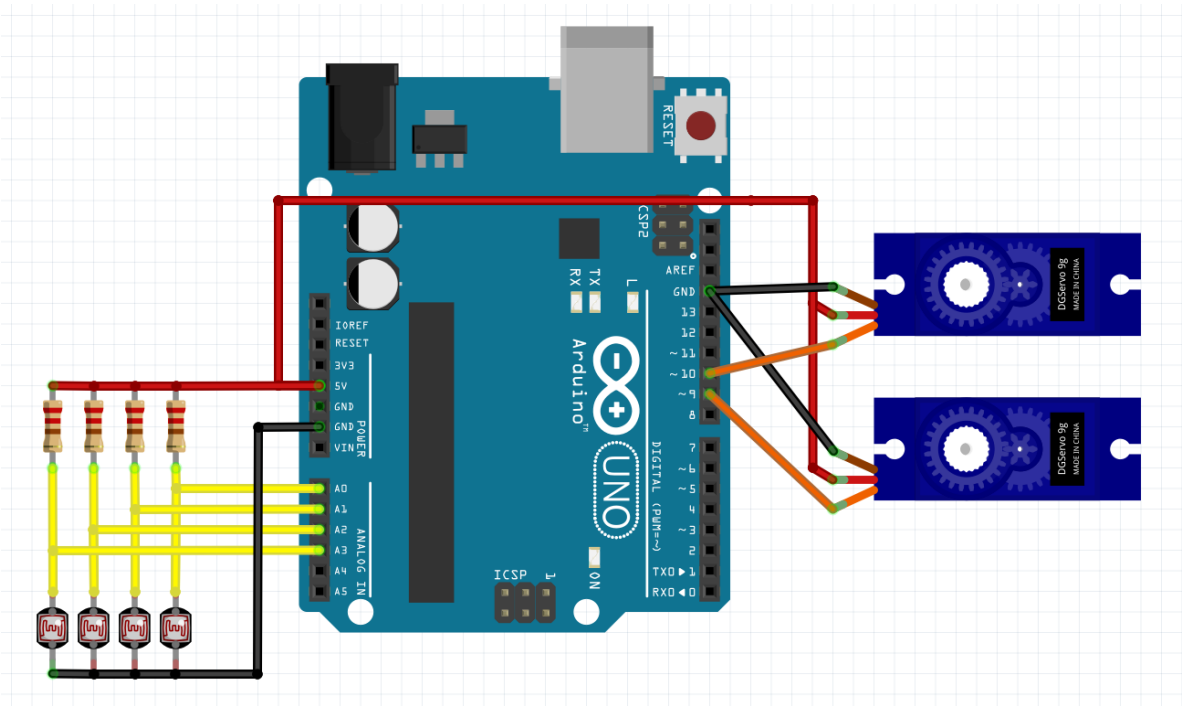
Beschreibung

Das Ziel ist es eine Apperatur zu erstellen die dem Sonnenlicht folgen kann und somit die Effizenz einer Solarzelle maximiert. Dafür steht eine um 2 Achsen bewegliche **Basis** zur Verfügung, die aus 3 Teilen und 2 Servos zusammengebaut wird.
Über 4 Lichtempfindliche Widerstände (LDR) sollen unsere Signale für die Servos ermittelt werden. Die LDR's werden in dem dazugehörigen **LDR Halter** montiert.
Zuletzt muss die Solarzelle sowie der LDR Halter auf der Basis befestigt werden.

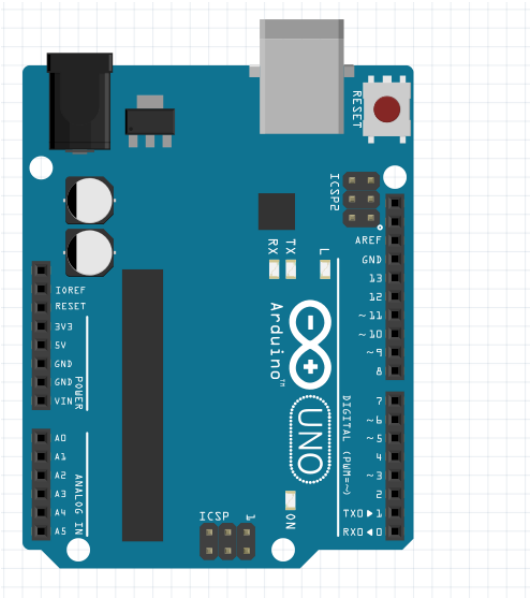
Komponenten

- 2x Servo
- Basis (3D Druck)
  - Pan-tilt-bottom.stl
  - Pan-tilt-middle.stl
  - Pan-tilt-top.stl
- 4x LDR (Photowiderstand)
- 4x Widerstand 1kΩ
- LDR Halter (3D Druck)
  - LDR-holder.stl
- 1x Solarzelle

Theorie



7 Project: Arduboy



Beschreibung

Eine weitere Methode, unsere LED zu betreiben, heißt Pulse Wide Modulation oder kurz **PWM**. Dies gibt uns die Möglichkeit, die Helligkeit zu regulieren. Dies funktioniert durch sehr schnelles Ein- und Ausschalten der LED. Wenn wir die "on time" verlängern, erreichen wir eine höhere Helligkeit. Wir benutzen ein Potentiometer als Eingang.
Die Pins, die diesen Modus verwenden können, sind mit ~ gekennzeichnet.

Komponenten

- 5x LED
- 5x Widerstand
- 1x Schalter

Theorie

Neuer Code