Übungen mit Tastern, Lichtschranken und Schallsensoren

Kann ein Sensor ***nur zwei Zustände*** annehmen (sinngemäß an und aus), so spricht man von einem ***digitalen Sensor***. So kann ein Taster gedrückt sein oder nicht, eine Lichtschranke „sieht“ ein Hindernis oder nicht … In solchen Fällen verwenden wir am Arduino entsprechend die ***digitalen Pins***.

Digitale Pins brauchen immer ein klares Signal: HIGH (5V) oder LOW (GND, 0V). Alle anderen Eingangs-Werte (Spannungen zwischen 0 und 5V) oder auch gar kein Eingangs-Singal können zu chaotischen, zufälligen Effekten führen und sind unbedingt zu vermeiden.

Deshalb hat man in der Digital-Technik folgendes vereinbart:

- ***HIGH*** bedeutet: ***„Es ist nichts los“***
- ***LOW*** bedeutet: ***„Der Sensor hat etwas wahrgenommen“***

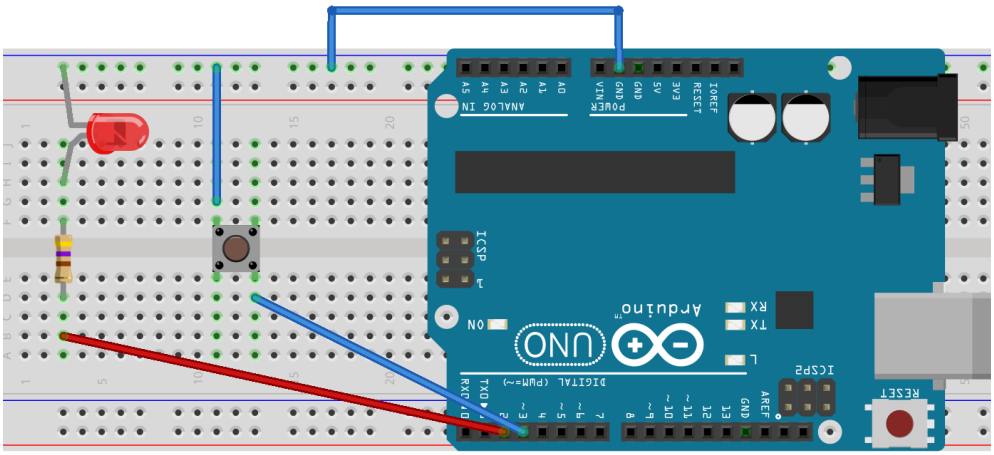
Deshalb sendet ein digitaler Taster ein HIGH, wenn er nicht gedrückt ist und ein LOW, wenn er gedrückt wird. Die Lichtschranke sendet ein HIGH, wenn sie kein Hindernis sieht und ein LOW, wenn sie ein Hindernis sieht.

Digitaler Taster und LED

Hardware

Du benötigst: **1 Taster**
1 LED
1 Widerstand *(250Ω – 1kΩ)*
1 Steckbrett
1 Arduino (Uno)

Aufbau

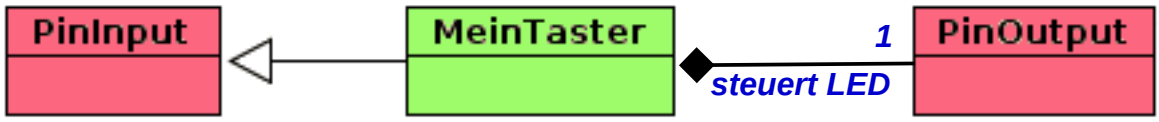


Achte darauf, dass die LED richtig herum angeschlossen ist.

Der Taster muss so eingebaut werden, dass die „Knie“ der Beine nach links und rechts und nicht nach oben und unten zeigen. Beim Tastendruck schaltet er dann nach unten durch und verbindet damit den GND (LOW) vom oberen Kabel mit dem Eingangs-Pin des Arduino.

Anforderungen an das Programm

Die ***Klasse PinInput*** benutzt man am besten, indem eine ***eigene Klasse davon erbt***. Man muss dann nur die geerbten Methoden `onReceiveChange()` , `onReceiveHigh()` , `onReceiveLow()` überschreiben und dort angeben, was bei dem jeweiligen Ereignis passieren soll.



1. Schreibe eine ***Klasse MeinTaster***, der von PinInput erbt.
   Rufe im Konstruktor den Super-Konstruktor mit zwei Übergabe-Parametern auf und übergib ihm die Pin-Nummer 2 und ein `true`. Das `true` sorgt dafür, dass der Pin ein High erhält, wenn der Taster nicht gedrückt ist. Ansonsten bekäme er in diesem Fall kein Signal :-(

   Tipp: *Sieh dir die Klassenkarte der Methode PinInput an und klicke dort auf den entsprechenden Konstruktor. Es erscheint eine detailliertere Hilfe.*

2. ***Überschreibe die Methoden*** `onReceiveChange()` und `onReceiveHigh()` mit leerem Rumpf. (Wir wollen auf diese Ereignisse keine Reaktion haben.)
   Überschreibe die Methode `onReceiveLow()` und gib in ihrem Rumpf aus, dass der Taster gedrückt wurde: `System.out.println("Taster gedrückt");`

3. Erzeuge nun ein Objekt deiner Klasse und lasse dir neben dem BlueJ-Hauptfenster das Konsolen-Fenster anzeigen. Teste, ob dein Taster auf Tasten-Druck reagiert.

   Es kann passieren, dass ein Tasten-Druck mehrere HIGH-LOW-Wechsel verursacht und damit mehrere Ausgaben auf der Konsole erzeugt. Man nennt diesen Effekt ***„prellen“*** *… das ist nichts ungewöhnliches.*
   Sollte dies passieren, so rufe im Konstruktor deiner Klasse MeinTaster nach dem Super-Konstruktor zusätzlich die Methode `super.setDebounce(5)` *auf. Damit sorgst du dafür, dass nach einem registrierten Tasten-Druck die nächsten 5 MilliSekunden nicht auf weitere Tasten-Drucke reagiert wird. Das prellen wird damit ignoriert.*

4. Damit der Taster etwas sinnvolleres tun kann, bringen wir die LED ins Spiel.
   Lege in deiner Klasse MeinTaster ein ***Referenz-Attribut*** vom ***Typ PinOutput*** an und nenne es `led`. Erzeuge das Objekt am Ende deines Konstruktors an Pin 2.
   Ändere nun den Rumpf der überschriebenen Methode `onReceiveLow()`:
   Nimm den alten Befehl weg und ersetze ihn durch `this.led.toggle()` .
   Damit schaltest du die LED an, wenn sie aus war und umgekehrt.

5. Erzeuge wieder ein Objekt deiner Klasse und Teste, ob alles funktioniert.

Klatsch-Licht

Eine witzige Alternative zum Taster mit LED ist ein Schall-Sensor mit LED. Klatscht man in die Hände oder ruft laut, so geht das Licht an, beim nächsten lauten Geräusch geht das Licht wieder aus.

Lies dir zunächst das Dokument

Schallsensor.pdf

aufmerksam durch.

6. Ersetze den Taster durch den Schallsensor.
   Da der Schallsensor sowieso mehrere HIGH-LOW-Wechsel ausgibt, ist es egal, in welcher der drei geerbten Methoden du die LED umschaltest.
   Denke im Konstruktor deiner Klasse allerdings daran, das Debounce-Intervall auf etwa 100-200 MilliSekunden einzustellen, wie es in der Anleitung zum Schallsensor beschreiben steht.

   Denkfrage: *Was könnte passieren, wenn du das Debounce-Intervall auf Null belässt?*

Lichtschranke und Summer

Ähnlich einfach kannst du eine Lichtschranke statt des Tasters oder Schallsensors verwenden. Noch witziger ist dann z.B. ein Summer statt einer LED.

Als Summer verwendest du am einfachsten einen sogenannten aktiven Piezo-Summer. Er hat zwei Beinchen, ein langes für den Plus-Pol und ein kurzes für den Minus-Pol. Schließt man ihn an, so gibt er einen sehr nervigen Ton von sich. Man kann ihn also genau wie die LED anschließen :-)

Lies dir nun das Dokument

Lichtschranke_Infrarot.pdf

aufmerksam durch.

7. Ersetze den Taster bzw. den Schallsensor durch die Lichtschranke.
   Ersetze außerdem die LED durch den Summer. Achte auf die richtige Polung des Summers!
   Überschreibe die drei geerbten Methoden so, dass der Summer angeht, wenn die Lichtschranke ein Hindernis sieht und wieder ausgeht, wenn das Hindernis wieder außerhalb der Reichweite der Lichtschranke ist.