Een schakelaar gebruiken

Arduino-experiment	130115-EN	*Trefwoorden*	Booleaanse uitdrukkingen, if, else, digitalRead(), opmerkingen		
Versie	2018-06-27 / HS	*Niveau*	Basiscursus, module 3	*p.*	1/4

Wat je leert:

- Lees de status van een input uit met `digitalRead()`
- Berekenen met true en false
- Beheers de programma-flow via voorwaarden
- Gebruik pull-up-weerstanden
- Switch bounce verhelpen

1 – Sluit een schakelaar aan op het breadboard

Drukknoppen

Drukknoppen (tactiele schakelaars) zijn er in veel verschillende maten en soorten. Plaats de schakelaar in een positie zoals hieronder aangegeven. Het type schakelaar dat wordt weergegeven heeft 4 pins, die twee aan twee zijn aangesloten. Geplaatst zoals afgebeeld, verbindt het de rijen aan weerszijden van de “opening” in het midden van het breadboard – we maken hier in de huidige opstelling geen gebruik van, maar het is goed om je er bewust van te zijn.

Als de opstelling niet werkt moet je de schakelaar met een ohmmeter controleren (zonder enige verbinding met de Arduino). De schakelaar zou moeten verbinden met rij 7 en rij 9 als hij wordt ingedrukt (weerstand van een paar ohm). Wanneer de knop wordt losgelaten, moet de verbinding worden verbroken (weerstand van veel MegaOhm of overflow op de meter).

Programmering

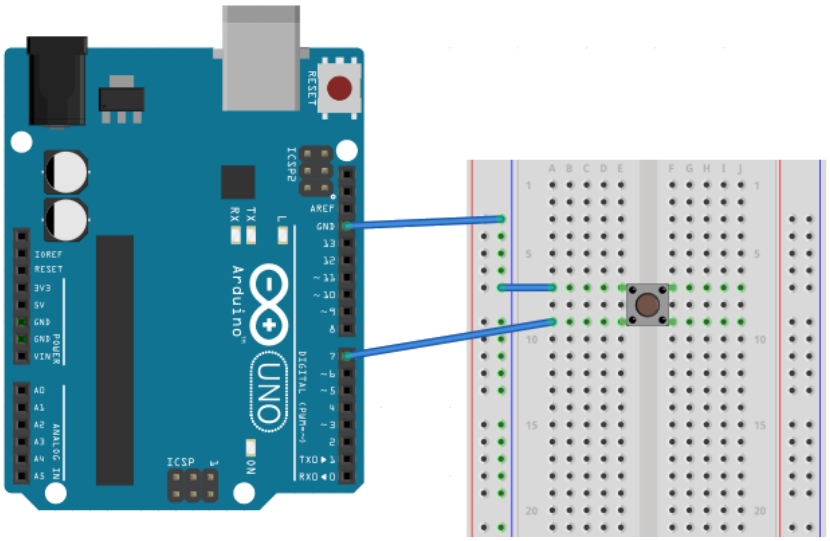
Voer het volgende programma in:

```
const byte pinLed = 13;
const byte pinSwitch = 7;

void setup() {
  pinMode(pinLed, OUTPUT);
  pinMode(pinSwitch, INPUT_PULLUP);
}

void loop() {
  if ( digitalRead(pinSwitch)==LOW ) {  // Button pressed
    digitalWrite(pinLed, HIGH);
  } else {
    digitalWrite(pinLed, LOW);
  }
}
```

Als alles werkt, moet de ingebouwde LED aan gaan terwijl de schakelaar wordt ingedrukt. Het programma wordt uitgelegd op pagina 2.



Het eerste nieuwe detail zit in `setup()` met de regel `pinMode (pinSwitch, INPUT_PULLUP);` Van pin 7 wordt een input gemaakt en een weerstand wordt intern aangesloten, waarbij geprobeerd wordt de ingang naar 5 V (=`HIGH`) te trekken. Dit wordt een pull-up-weerstand genoemd. De weerstand is zo hoog dat de schakelaar zonder problemen de input naar beneden kan trekken tot 0 V (=`LOW`) als de knop wordt ingedrukt.

Het volgende detail dat opvalt is de constructie

```
if ( … ) { … } else { … }
```

Deze 5 regels werken als volgt: Als de toets wordt ingedrukt, moet de LED branden – anders moet deze uit gaan.

De toestand in de ronde haakjes (...) gebruikt de functie `digitalRead()` om de toestand van de input-pin `pinSwitch` uit te lezen en te vergelijken met de constante `LOW`.

Het dubbele gelijkheidsteken "==" moet worden gelezen als "is gelijk aan" – in tegenstelling tot "=" dat luidt: "krijgt de waarde van".

De `if`-constructie is gedetailleerd en samengevat in de toolbox rechts.

Het derde detail om op te merken is de *opmerking*:

```
// Button pressed
```

Alles van "//" tot het einde van de regel is een opmerking en heeft geen gevolgen voor de uitvoering van het programma. Maar voor mensen die de code proberen te lezen, zijn goede opmerkingen zeer waardevol om het programma te begrijpen.

Een andere manier om een opmerking te maken is door gebruik te maken van de begin- en eindmarkeringen `/*` en `*/`. Dit soort commentaar kan meerdere regels beslaan.

Toolbox: Voorwaardelijke uitvoering

Voorbeeld:

```
if (i == 4) {
  digitalWrite(pinLed, HIGH);
} else {
  digitalWrite(pinLed, LOW);
}
```

Als de variabele `i` de waarde 4 heeft, brandt de LED, anders gaat deze uit.

Uitleg:

```
if ( A ) { B } else { C }
```

A Een uitdrukking die waar (true) of onwaar (false) is

B Instructies uitgevoerd indien ***A*** waar is

C Instructies uitgevoerd indien ***A*** onwaar is

Als ***B*** of ***C*** slechts bestaan uit een enkele instructie, kunnen de accolades **{** en **}** worden weggelaten.

Mocht er niets gebeuren wanneer ***A*** onwaar is, kun je `else` en `{ C }` weglaten.

Voorbeeld 2:

```
if (i==4) digitalWrite(pinLed,HIGH);
```

Als de variabele `i` de waarde 4 heeft, brandt de LED.

2 – Reageren op een ingedrukte schakelaar

Eén taak per schakelaar

Uitdaging 1

Verander het programma zodat de schakelaar de LED aan laat gaan en deze dan 2 seconden blijft branden, zelfs als de knop wordt losgelaten – en daarna weer uitschakelt.

(Als de knop ingedrukt wordt gehouden, moet de LED onmiddellijk weer gaan branden.)

Uitdaging 2

Sluit een andere schakelaar aan op de Arduino, bijvoorbeeld op pin 6.

Voeg een regel toe in `setup()`, waardoor deze pin ook een input is met een pull-up-weerstand.

De LED moet niet langer worden aangestuurd door de tijd, maar moet aan gaan als de eerste knop wordt ingedrukt en uit gaan door de andere knop in te drukken.

Meer taken voor een enkele schakelaar

Tot nu toe hadden de schakelaars elk één – zeer beperkte – taak. Vaak zal een knop meerdere functies moeten uitvoeren, zoals het doorlopen van een reeks menu-items, een nieuwe voor elke druk op de knop.

We kiezen een heel eenvoudig voorbeeld: éénmaal drukken moet de LED aanzetten, nogmaals drukken moet de LED uitzetten – etc. Je zult tijdens deze oefening een aantal interessante details ontdekken.

We hebben meteen al deze vragen om over na te denken:

Hoe weet het programma of de LED aan of uit is?

Hoe weet het programma het verschil tussen een “nieuwe” druk op de knop en een verlengde druk?

De eerste poging om het probleem op te lossen zou er zo uit kunnen zien:

```
const byte pinLed = 13;
const byte pinSwitch = 7;
bool ledState=LOW;

void setup() {
  pinMode(pinLed,OUTPUT);
  pinMode(pinSwitch,INPUT_PULLUP);
}

void loop() {
  while(digitalRead(pinSwitch)==HIGH);

  ledState = !ledState;
  digitalWrite(pinLed,ledState);

  while(digitalRead(pinSwitch)==LOW);
}
```

Het programma houdt de toestand van de LED bij door middel van een *Booleaanse variabele* `ledState` (ook wel een *logische* variabele genoemd). Het begint met de waarde `LOW` wat hetzelfde is als `false`.

Zie de toolbox rechts.

De twee regels

```
ledState = !ledState;
digitalWrite(pinLed,ledState);
```

geeft de variabele `ledState` eerst de tegenovergestelde waarde, die vervolgens wordt gebruikt om de LED aan of uit te zetten.

Om ***één*** druk op de knop tegelijk te beheren, bevat het programma twee `while` loops. Zie de toolbox rechts.

De eerste loop

```
while(digitalRead(pinSwitch)==HIGH);
```

doet – niets(!) – *zolang de knop niet is ingedrukt*. Er is geen instructie na de haakjes met de loopvoorwaarde, alleen een puntkomma. Dit wordt een empty loop genoemd.

Zodra niet meer aan de voorwaarde wordt voldaan, moet dit betekenen dat de schakelaar is ingedrukt. Dit leidt tot de uitvoering van de bovenstaande regels die de toestand van de LED verandert.

Dan bereiken we de onderste `while` loop:

```
while(digitalRead(pinSwitch)==LOW);
```

Het programma blijft in deze loop zolang *de schakelaar nog ingedrukt is*. Als de knop wordt losgelaten, gaat het programma verder met de volgende ronde in de `loop()`-functie – en eindigt weer in de eerste `while` loop.

Dit kan haast niet misgaan, toch?

Toolbox: Logische (Booleaanse) variabele

Voorbeeld:

```
bool itIsSeven;
int j = 5;
itIsSeven = (j == 7);
```

De variabele `itIsSeven` kan de waarden `false` of `true` hebben. Aangezien de integer-variabele `j` 5 is, is de uitdrukking `(j == 7)` false. Na de derde regel is de waarde van `itIsSeven` daarom `false` .

Booleaanse uitdrukkingen kunnen worden gemaakt met de operators `&&` (*and,* en), `||` (*or,* of), en `!` (*not,* niet) .

Voorbeeld:

```
ledState = !ledState;
```

Deze regel geeft `ledState` de *tegenovergestelde*

Toolbox: Herhalingsloop met `while`

Voorbeeld:

```
int sum = 0;
int j = 0;
while (j<11) {
  sum = sum + j
  j++;
}
```

Dit stukje code berekent de som van de integers van 0 tot 10.

Uitleg:

`while ( A ) { B }`

A De voorwaarde ***A*** wordt geëvalueerd **vóór elke** uitvoering van de loop. Als de uitdrukking true is, wordt ***B*** uitgevoerd.(Anders springt het programma naar de eerste instructie na de while-loop)

B Eén of meer instructies. Als er maar één is, kunnen de accolades { } weggelaten worden. – De instructie kan zelfs weggelaten worden, waardoor alleen de puntkomma overblijft.

Uitdaging 3

Tijd voor een proefrit van het programma!

Controleer zorgvuldig of de LED na *elke* druk op de knop verandert. Evenals of de LED elke keer dat je twee keer hebt gedrukt in dezelfde toestand terechtkomt.

3 – Debouncing

Switch bounce

Wat was het resultaat van je test in *Uitdaging 3*?

Tenzij je het geluk hebt om een aantal zeer speciale schakelaars te hebben, zou je moeten concluderen dat het werkt – *af en toe*! Maar vaak zie je dat de LED slechts een kleine flits geeft op de plaats waar hij aan had moeten gaan. Op andere momenten lijkt er geen enkele reactie te zijn.

Je hebt het fenomeen *switch bounce* ontdekt:
Als je de schakelaar één keer indrukt, stuiteren de metalen onderdelen in de schakelaar een paar keer terug voordat ze zich settelen. En als je de knop loslaat, slepen de metalen onderdelen tegen elkaar aan en maken ze weer contact nadat het contact verbroken was. Zelfs als *je* slechts één keer drukt, wordt er vele malen contact gemaakt.

- Hoe kunnen we dit verhelpen?

Software debouncing

Er zijn hier verschillende technieken beschikbaar. Wij kiezen voor een zeer eenvoudige softwareoplossing:

Dit probleem is beperkt tot een zeer korte tijd direct na het indrukken of loslaten van de schakelaar. Je kunt het programma dus gewoon vragen om een pauze te nemen, direct na elk van de twee `while` loops. Gebruik hiervoor `delay()`.

Uitdaging 4

Voeg zoals gezegd de regels met `delay()` toe. Definieer een constante `debounceTime` voor de wachttijd voor het aanroepen van de `delay()`-functie. Probeer het eerst met zo'n 50 milliseconden.

Test het programma.

Probeer `debounceTime` aan te passen – hoe weinig is er eigenlijk nodig? Het programma moet zich nog steeds volledig stabiel gedragen!

Elke techniek die de gevolgen van switch bounce verhelpt, wordt *debouncing* genoemd. Dit kan zowel in software worden gedaan als in hardware, waar de schakelingen worden aangepast.