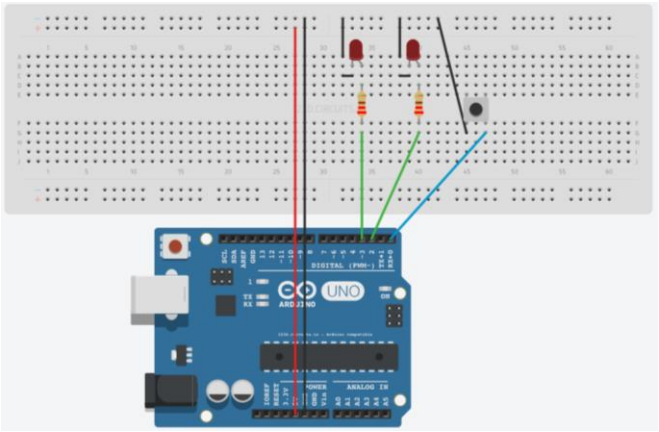
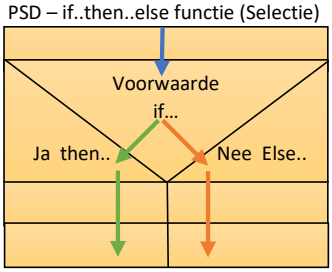
Keuzestructuren

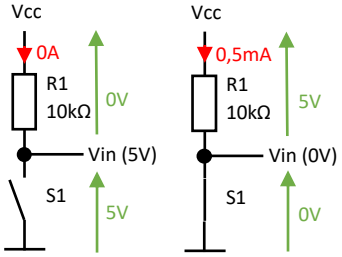



```
void setup() {

   Serial.begin(9600);                 //start seriële verbinding

   pinMode(0, INPUT_PULLUP);           //configureer pin0 als een input en
                                       //enable de interne pull-up weerstand
   pinMode(2, OUTPUT);                 //configureer pin2 als een output
   pinMode(3, OUTPUT);                 //configureer pin3 als een output
}
void loop() {

   int sensorVal = digitalRead(0);     //lees de waarde van de drukknop en zet deze in de
                                       //variabele sensorVal

   Serial.println(sensorVal);          //Stuur de waarde van de drukknop (sensorVal)
                                        //naar de seriële monitor

// Door de pullup-weerstand reageert de uitgang geinverteerd op de stand van de schakelaar
// De uitgang is HOOG (logisch "1") als de schakelaar open is.
// De uitgang is LAAG (logisch "0") als de schakelaar dicht is.

// if..then..else functie
   if (sensorVal == HIGH) {            //Als de variabele sensorVal hoog is dan
     digitalWrite(2, LOW);             // Zet uitgang 2 LAAG
     digitalWrite(3, HIGH);            // Zet uitgang 3 HOOG
  }
     else {                            //Anders
     digitalWrite(2, HIGH);            // Zet uitgang 2 HOOG
     digitalWrite(3, LOW);             // Zet uitgang 3 LAAG
   }
}
```

PSD – if..then..else functie (Selectie)



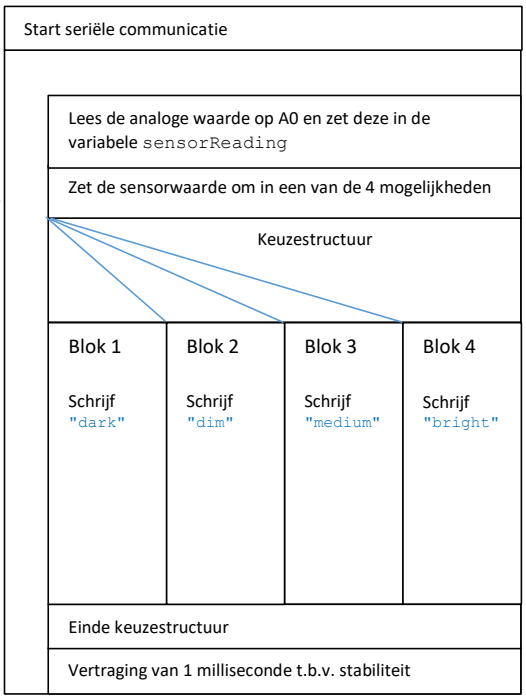
Pull up weerstand



Een ingang mag nooit zweven (los hangen). Op een loshangende ingang kan door straling een HOOG-signaal (logische "1") komen te staan. Een manier om dit te voorkomen is een pull-up weerstand.
Als in het schema hierlangs de schakelaar open staat, dan is de spanning Vi = 5V. Vi zit vast aan een ingang die een hele hoge weerstand heeft. Hierdoor loopt er geen stroom door de weerstand. U = I * R -> U = 0A * 10.10^3 = 0V. Er is dus geen spanningsval over de weerstand.
Als de schakelaar gesloten wordt hangt Vi direct aan de massa. Vi wordt dan 0V. Er gaat dan wel een stroom lopen door de weerstand:
I = U/R = 5V/10.10^3 =$0,5.10^{-3}$ = 0,5 mA

```
/*
  State change detection (edge detection)

 Often, you don't need to know the state of a digital input all the time,
 but you just need to know when the input changes from one state to another.
 For example, you want to know when a button goes from OFF to ON.  This is called
 state change detection, or edge detection.

 This example shows how to detect when a button or button changes from off to on
 and on to off.

 The circuit:
 * pushbutton attached to pin 2 from +5V
 * 10K resistor attached to pin 2 from ground
 * LED attached from pin 13 to ground (or use the built-in LED on
   most Arduino boards)

 http://www.arduino.cc/en/Tutorial/ButtonStateChange

 */

// this constant won't change:
const int  buttonPin = 2;                 // the pin that the pushbutton is attached to
const int ledPin = 13;                    // the pin that the LED is attached to

// Variables will change:
int buttonPushCounter = 0;                // counter for the number of button presses
int buttonState = 0;                      // current state of the button
int lastButtonState = 0;                  // previous state of the button

void setup() {

  pinMode(buttonPin, INPUT);              // initialize the button pin as a input:
  pinMode(ledPin, OUTPUT);                // initialize the LED as an output:
  Serial.begin(9600);                     // initialize serial communication:
}

void loop() {

  buttonState = digitalRead(buttonPin);   //lees de waarde van de schakelaar op ingang
                                          //buttonPin (ingang 2)

  if (buttonState != lastButtonState) {   //Als buttonState ongelijk(!=) is aan lastButtonState
    if (buttonState == HIGH) {            //Als buttonState gelijk is aan (==) HOOG dan,
      buttonPushCounter++;                //buttonPushCounter met 1 verhogen (++ -> x = x + 1)
      Serial.println("on");               //schrijf “on”
      Serial.print("number of button pushes:  ");
      Serial.println(buttonPushCounter);
    } else {                              //Als buttonState gelijk is aan (==) LAAG dan,
      Serial.println("off");              //schrijf “off”
    }

    delay(50);                            //Vertraging van 50ms om trillingen van de
                                          //schakelaar te vermijden
  }

  lastButtonState = buttonState;          //lastButtonState krijgt de waarde van buttonState
                                          //for next time through the loop

  // turns on the LED every four button pushes by
  // checking the modulo of the button push counter.
  // the modulo function gives you the remainder of
  // the division of two numbers:

  if (buttonPushCounter % 4 == 0) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }

}
```

```
/*
  Switch statement

 Demonstrates the use of a switch statement.  The switch
 statement allows you to choose from among a set of discrete values
 of a variable.  It's like a series of if statements.

 To see this sketch in action, but the board and sensor in a well-lit
 room, open the serial monitor, and and move your hand gradually
 down over the sensor.

 The circuit:
 * photoresistor from analog in 0 to +5V
 * 10K resistor from analog in 0 to ground

 http://www.arduino.cc/en/Tutorial/SwitchCase
 */

// these constants won't change. They are the
// lowest and highest readings you get from your sensor:
const int sensorMin = 0;
// sensor minimum, experimenteel vastgesteld
const int sensorMax = 600;
// sensor maximum, experimenteel vastgesteld

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
}
void loop() {
  // read the sensor:
  int sensorReading = analogRead(A0);
  // map the sensor range to a range of four options:
  int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

  // do something different depending on the
  // range value:
  switch (range) {
  case 0:    // your hand is on the sensor
    Serial.println("dark");
    break;
  case 1:    // your hand is close to the sensor
    Serial.println("dim");
    break;
  case 2:    // your hand is a few inches from the sensor
    Serial.println("medium");
    break;
  case 3:    // your hand is nowhere near the sensor
    Serial.println("bright");
    break;
  }
  delay(1);        // delay in between reads for stability
}
```



`map` commando
https://www.arduino.cc/en/Reference/Map

Beschrijving

De map() functie een getal om van het ene bereik naar een ander bereik (scaleren of verschalen)

map(value, fromLow, fromHigh, toLow, toHigh)

Value = waarde

fromLow = oude lage waarde

fromHigh = oude hoge waarde

toLow = nieuwe lage waarde

toHigh = nieuwe hoge waarde

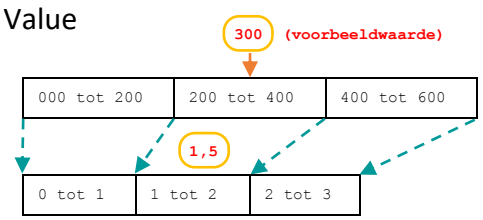
Voorbeeld_1 - verschalen

map(value, fromLow, fromHigh, toLow, toHigh)

map(300 , 0 , 600 , 0 , 3)

fromLow	fromHigh
0 (=sensorMin)	600 (=sensorMax)

toLow	toHigh
0	3



Voorbeeld_2 - Inverteren

Je kunt de map() functie ook gebruiken voor een serie getallen te inverteren:

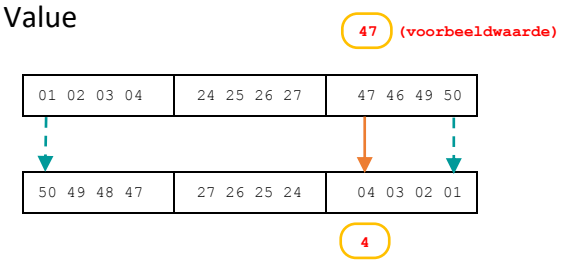
```
y = map(x, 1, 50, 50, 1);
```

map(value, fromLow, fromHigh, toLow, toHigh)

map(x , 1 , 50 , 50 , 1)

fromLow	fromHigh
1 (=sensorMin)	50 (=sensorMax)

toLow	toHigh
50	1



Uitwerking

```
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01
```

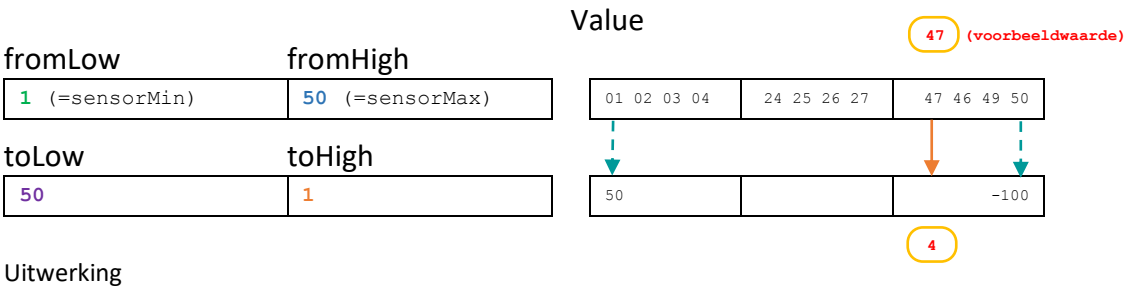
Voorbeeld_3 - Inverteren

De map() functie werkt ook met negatieve getallen:

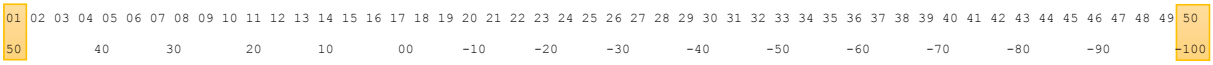
```
y = map(x, 1, 50, 50, -100);
```

map(value, fromLow, fromHigh, toLow, toHigh)

map(x , 1 , 50 , 50 , -100)



Uitwerking



In bovenstaand voorbeeld zouden er bij het omzetten geen hele getallen uitkomen.

De map() functie maakt echter alleen gebruik van integers (gehele getallen). Er ontstaan dus geen komma-getallen. Mocht er bij het omzetten een komma-getal ontstaan wordt deze afgekapt bij de komma en niet afgerond of gemiddeld. Voorbeelden: een 8,4 wordt een 8, een 8,9 wordt ook een 8.

.

The constrain() function

State Change Detection (Edge Detection) for pushbuttons

Zodra je een pushbutton werkend hebt, kun je ook een schakeling maken waarin je kunt tellen hoe vaak een button is ingedrukt. Hiervoor moet je weten wanneer de button van het “0” naar “1” niveau verandert, de zogenaamde opgaande flank. Deze kun je dan tellen.

Opgaande flank

Once you've got a working, you often want to do some action based on how many times the button is pushed. To do this, you need to know when the button changes state from off to on, and count how many times this change of state happens. This is called state change detection or edge detection. In this tutorial we learn how to check the state change, we send a message to the Serial Monitor with the relevant information and we count four state changes to turn on and off an LED.

Hardware Required

- Arduino or Genuino Board
- momentary button or switch
- 10k ohm resistor
- hook-up wires
- breadboard

Circuit

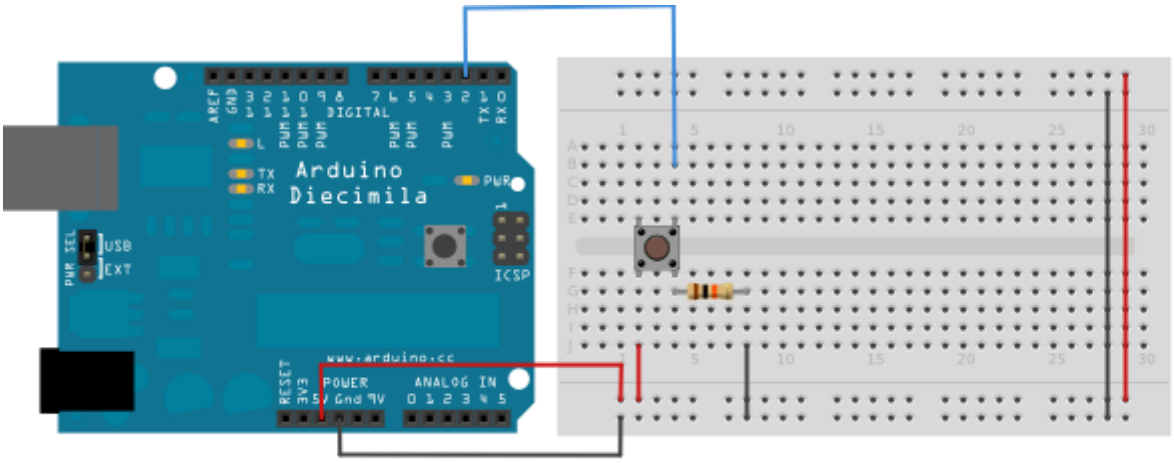


image developed using Fritzing. For more circuit examples, see the Fritzing project page

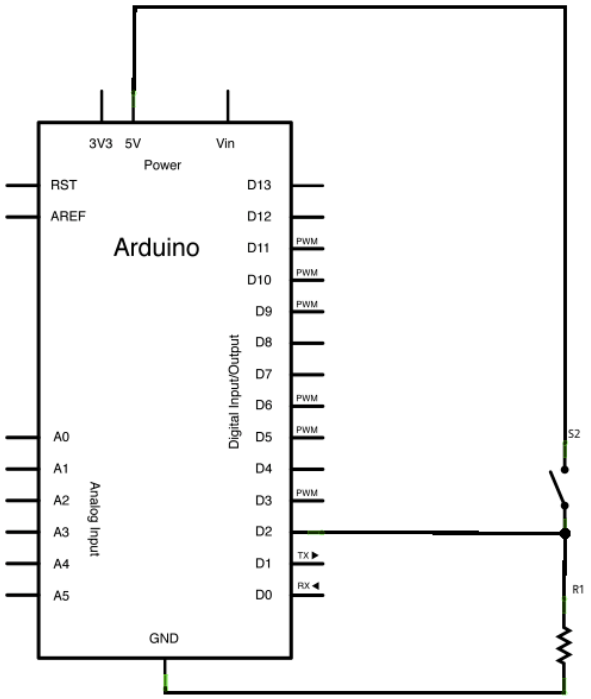
Connect three wires to the board. The first goes from one leg of the pushbutton through a pull-down resistor (here 10k ohm) to ground. The second goes from the corresponding leg of the pushbutton to the 5 volt supply. The third connects to a digital I/O pin (here pin 2) which reads the button's state.

When the pushbutton is open (unpressed) there is no connection between the two legs of the pushbutton, so the pin is connected to ground (through the pull-down resistor) and we read a LOW. When the button is closed (pressed), it makes a connection between its two legs, connecting the pin to voltage, so that we read a HIGH. (The pin is still connected to ground, but the resistor resists the flow of current, so the path of least resistance is to +5V.)

If you disconnect the digital I/O pin from everything, the LED may blink erratically. This is because the input is "floating" - that is, not connected to either voltage or ground. It will more or less randomly return either HIGH or LOW. That's why you need a pull-down resistor in the circuit.

Schematic

click the image to enlarge



Code

The sketch below continually reads the button's state. It then compares the button's state to its state the last time through the main loop. If the current button state is different from the last button state and the current button state is high, then the button changed from off to on. The sketch then increments a button push counter.

The sketch also checks the button push counter's value, and if it's an even multiple of four, it turns the LED on pin 13 ON. Otherwise, it turns it off.

```
/*
  State change detection (edge detection)

 Often, you don't need to know the state of a digital input all the
```

```
time,
 but you just need to know when the input changes from one state to
another.
 For example, you want to know when a button goes from OFF to
ON.  This is called
 state change detection, or edge detection.

 This example shows how to detect when a button or button changes
from off to on
 and on to off.

 The circuit:
 * pushbutton attached to pin 2 from +5V
 * 10K resistor attached to pin 2 from ground
 * LED attached from pin 13 to ground (or use the built-in LED on
   most Arduino boards)

 created  27 Sep 2005
 modified 30 Aug 2011
 by Tom Igoe

This example code is in the public domain.

 http://www.arduino.cc/en/Tutorial/ButtonStateChange

 */

// this constant won't change:
const int  buttonPin = 2;    // the pin that the pushbutton is
attached to
const int ledPin = 13;       // the pin that the LED is attached to

// Variables will change:
int buttonPushCounter = 0;   // counter for the number of button
presses
int buttonState = 0;         // current state of the button
int lastButtonState = 0;     // previous state of the button

void setup() {
  // initialize the button pin as a input:
  pinMode(buttonPin, INPUT);
  // initialize the LED as an output:
  pinMode(ledPin, OUTPUT);
  // initialize serial communication:
  Serial.begin(9600);
}


void loop() {
  // read the pushbutton input pin:
  buttonState = digitalRead(buttonPin);
```

```
  // compare the buttonState to its previous state
  if (buttonState != lastButtonState) {
    // if the state has changed, increment the counter
    if (buttonState == HIGH) {
      // if the current state is HIGH then the button
      // wend from off to on:
      buttonPushCounter++;
      Serial.println("on");
      Serial.print("number of button pushes:  ");
      Serial.println(buttonPushCounter);
    } else {
      // if the current state is LOW then the button
      // wend from on to off:
      Serial.println("off");
    }
    // Delay a little bit to avoid bouncing
    delay(50);
  }
  // save the current state as the last state,
  //for next time through the loop
  lastButtonState = buttonState;


  // turns on the LED every four button pushes by
  // checking the modulo of the button push counter.
  // the modulo function gives you the remainder of
  // the division of two numbers:
  if (buttonPushCounter % 4 == 0) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }

}
```