Handleiding Arduino Basis

Fun met Electronica

2018

Inhoud

Beschrijving Arduino is een opensource–computerplatform bedoeld om microcontrollers eenvoudig te programmeren. Dit platform is bedoeld voor hobbyisten, artiesten, kunstenaars en iedereen die geïnteresseerd is in het maken en ontwerpen van slimme en creatieve objecten. Arduino is uitermate geschikt voor projecten, prototypes en meetopstellingen. Vanwege de grote community, kan er van alles gevonden worden op het internet. Bijna alle mogelijke componenten voor de Arduino hebben tutorials en voorbeeldcode, waardoor het gebruik ervan enorm makkelijk is.

1 Inleiding elektronica

Fun met Electronica heeft twaalf basis elektronica projecten uitgewerkt. Er zal begonnen worden met een eenvoudige stroomkring met een led en schakelaars en geïndigd worden met een looplicht met transistoren en condensatoren. Van elk project is een elektrisch schema getekend en er is van elk project een Fritzing tekening gemaakt om het bouwen van de schakeling duidelijker te maken. Volg deze inleiding vooraf de training.

1.1 Benodigdheden

Alle projecten die hier beschreven worden kunnen worden uitgevoerd met de volgende sets:

- 'Electronica Experimenteerset Basis' artikel nummer 2101
- 'Electronica Experimenteerset Extra' artikel nummer 2102
- 'Funduino Experimenteerset Extra Compleet' artikel nummer 2212

Deze sets kunnen gevonden worden op de webshop van Fun met Electronica. Deze is te vinden op `shop.funmetelectronica.nl`. De inhoud van de Electronica Experimenteerset Basis is:

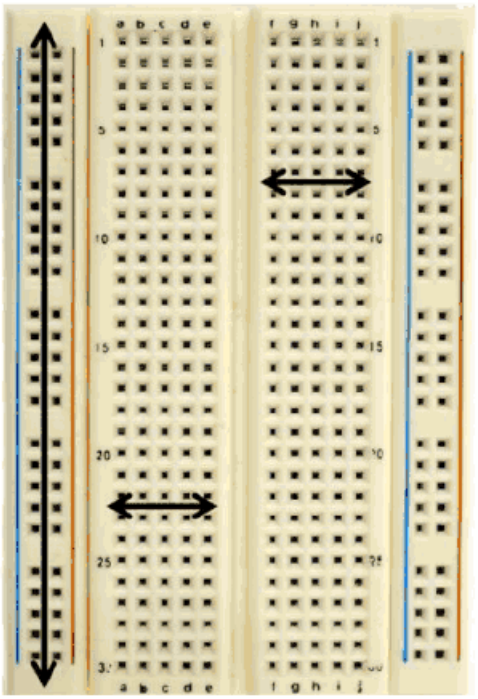
1. 1x Opberg box met deksel
2. 1x Breadboard 400
3. 10x Jumper wire male male
4. 1x 9V batterij connector met pinheader
5. 5x Switch
6. 5x led Rood 5mm
7. 5x led Groen 5mm
8. 5x Transistor BC 547
9. 5x Transistor BC 557
10. 2x Transistor BS 170
11. 2x Condensator 0,1 uF (100nF)
12. 6x Elektrolytische Condensator 10 uF
13. 2x Elektrolytische Condensator 100 uF
14. 2x LDR
15. 5x Weerstand 470 Ω 0,25W
16. 5x Weerstand 1 KΩ 0,25W
17. 5x Weerstand 10 KΩ 0,25W
18. 5x Weerstand 51 KΩ 0,25W
19. 5x Weerstand 100 KΩ 0,25W
20. 5x Weerstand 10 MΩ 0,25W

1.2 Onderdelen

1.2.1 Breadboard

Een breadboard is een gaatjesboard waar componenten ingestoken kunnen worden. De gaatjes van het breadboard zijn op een bepaalde manier met elkaar verbonden. Zonder te solderen, is het met een breadboard mogelijk om snel componenten met elkaar te verbinden. Als je klaar bent met een opstelling, haal je alles weer uit elkaar. De onderdelen zijn dan opnieuw te gebruiken. Hoe de gaatjes met elkaar zijn doorverbonden, is te zien aan de pijlen in Afbeelding 1.

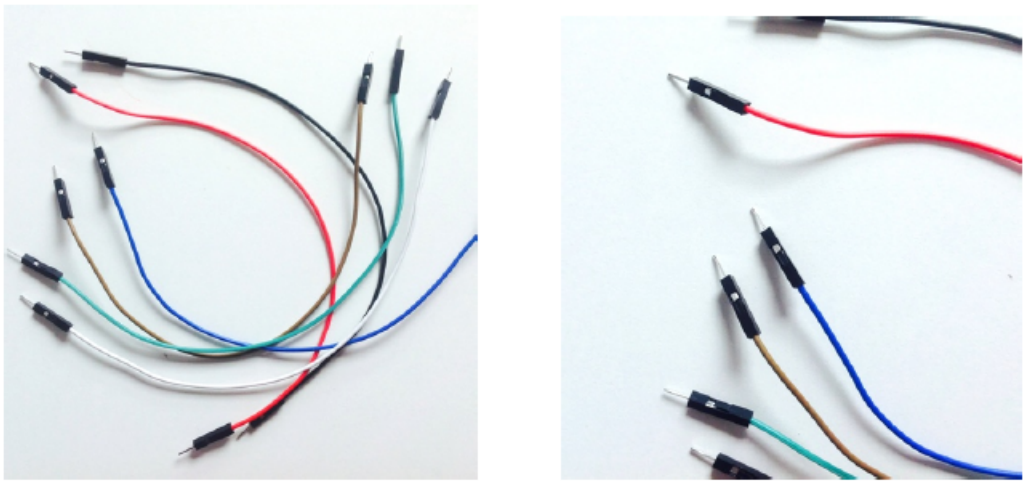
Voorbeeld: gaatje A1 is verbonden met D1, maar ook met B1, C1, en E1. A1 is niet verbonden met F1. Links en rechts zijn gaatjes verticaal met elkaar verbonden, hierop wordt doorgaans de voeding op aangesloten. Rood voor plus, 5 Volt of VCC en blauw voor min, GND, 0 Volt of VDD.



Afbeelding 1: Een breadboard

1.2.2 Jumper wires

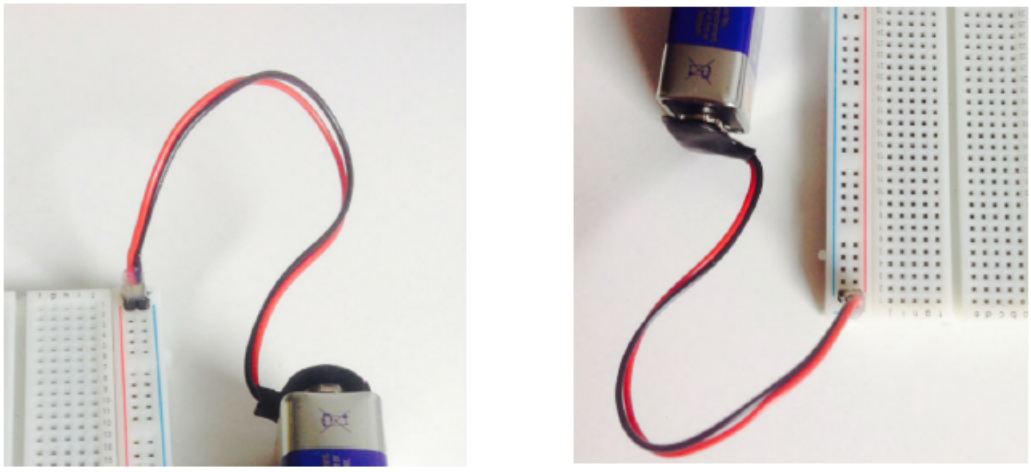
De jumper wires die in deze set worden meegeleverd zijn van het type 'male/male' (Afbeelding 2). De stekkertjes steek je eenvoudig in het breadboard. Zo kun je gemakkelijk de ene rij met de andere rij gaatjes verbinden. De jumper wires zijn er in verschillende kleuren. We proberen rood zo veel mogelijk voor de pluskant te gebruiken en zwart voor de minkant te gebruiken. Ze kunnen kapot gaan maar gelukkig zijn dit enorm goedkope componentjes.



Afbeelding 2: Jumper wires

1.2.3 Batterijen

Een 9 Volt batterij kan gebruikt worden als voeding voor de arduino. De gesoldeerde pinheader past op het breadboard zoals te zien op Afbeelding 3. Let op: rood bij rood, zwart bij blauw.



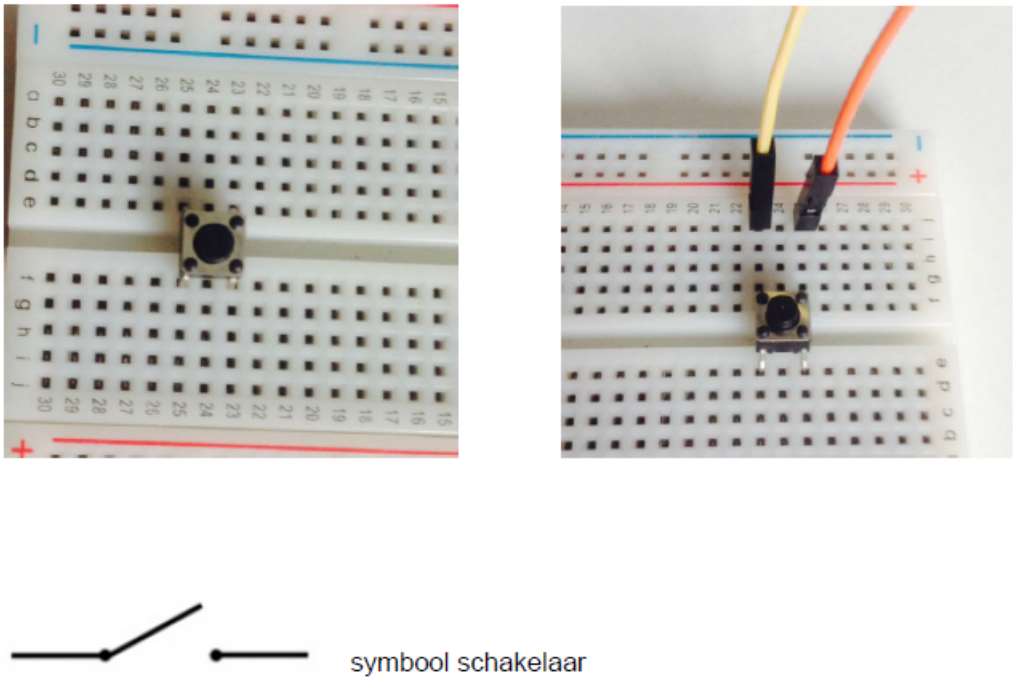
Afbeelding 3: Een batterij met jumper wires eraan gesoldeerd

In Afbeelding 4 zijn de symbolen van een batterij of voedingsbron te zien. Elke dikke streep stelt één cel voor. Elke cel in een batterij levert 1.5 Volt, een 9 Volt batterij heeft dus 6 cellen. Vaak worden niet alle lijnen getekend.

Afbeelding 4: Het schema van een voedingsbron en batterij

1.2.4 Switch

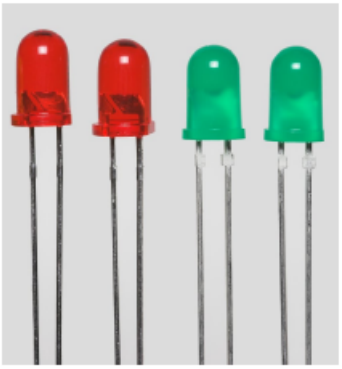
De switch, drukknop of schakelaar is een enkelpolige schakelaar waarvan de aansluitingen dubbel zijn uitgevoerd. De aansluitingen zijn in Afbeelding 5 links boven en onder met elkaar doorverbonden. Als op de knop gedrukt wordt, zullen links en rechts met elkaar verbonden worden.



Afbeelding 5: Een schakelaar die links en rechts met elkaar verbind

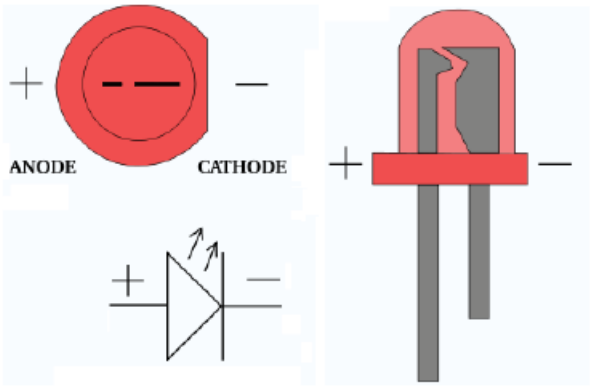
1.2.5 Light Emitting Diode (led)

Led staat voor Light Emitting Diode, oftewel een diode die licht uit straalt. Een diode laat de stroom maar in één richting door. Geeft de led geen licht, dan dient deze andersom aangesloten te worden. Geeft deze nog steeds geen licht, dan is de led wellicht kapot. Aan de pootjes herken je de pluskant en de minkant van de led. Lang is plus, kort is min.



Afbeelding 6: Een foto van twee leds

De aansluitspanning is afhankelijk van de kleur van de led. Een rode led heeft bijvoorbeeld 1.6 Volt nodig, een gele 1.8 Volt, een groene 2.2 Volt en blauwe en witte led 3.4 Volt. Als we een led aansluiten op een batterij van 4.5 of 9 Volt zonder weerstand, gaat de led stuk. De led brandt door omdat de stroom die door de led gaat te hoog is. We moeten een weerstand in de stroomkring gebruiken.



Afbeelding 7: Een schematekening van een led

1.2.6 Weerstand

Een weerstand bemoeilijkt de elektrische stroom in een stroomkring met als gevolg een spanningsval over een weerstand. Met de juiste weerstand kunnen we bijvoorbeeld een led aansluiten op een 9 volt batterij. We kunnen bij een bepaalde spanning een duidelijke stroom creëren, wat voor vele componenten zoals een led nodig is. Op Afbeelding 8 zie je de meest gebruikelijke symbolen voor weerstanden.

Afbeelding 8: Symbolen voor een weerstand

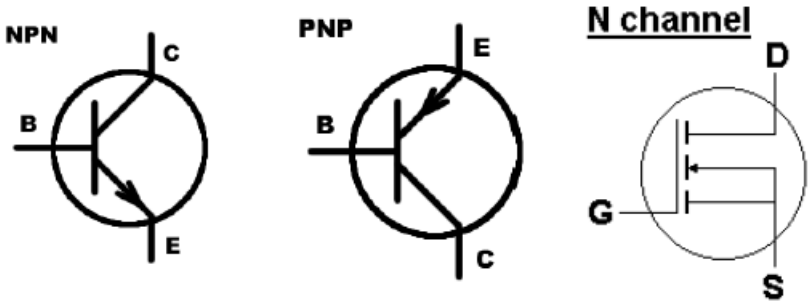
De grootte van de weerstand wordt weergegeven in Ohm (Ω). Met de gekleurde ringen kunnen we de weerstand bepalen. Hiervoor is een kleurcodetabel gemaakt. De weerstand in het Afbeelding 8 is: 1000 Ω. 1 (eerste ring is bruin) 0 (tweede ring is zwart) 00 (derde ring is rood). De vierde ring geeft de tolerantie van de weerstand aan. In werkelijkheid kan deze weerstand een beetje groter of kleiner zijn. Een goudkleurige ring staat voor 5% tolerantie. Een handige hulp voor het uitrekenen is weerstandcalculator. Er bestaan ook diverse Apps.

1.2.7 Transistoren

Een transistor wordt gebruikt om signalen te versterken of om automatisch te schakelen als een knop. Het versterken wordt op dit niveau nog niet besproken. Er zijn twee soorten transistoren: bipolaire transistoren en veldeffecttransistoren beiden met andere eigenschappen. De twee soorten hebben allebei 3 connecties en 2 types. Deze types zijn de NPN en de PNP, deze hebben de tegengestelde werking. In Afbeelding 9 zijn de schema's van deze transistoren getekend.

Bipolaire transistor. De aansluitingen heten de basis, collector en emitter. Bij een hoge spanning op over de collecor en emitter gaat de poort van de schakeling open en kan er een stroom lopen door de collector vanaf de basis.

Veldeffecttransistor. De aansluitingen heten de drain, gate en source. Bij een hoge spanning op de gate gaat de poort tussen de drain en source open.



Afbeelding 9: Symbolen voor een transistor

1.2.8 Condensatoren

Een condensator is een onderdeel die elektrische energie kan opslaan. In de projecten worden er twee soorten gebruikt. Een ceramische condensator en een elektrolytische condensatoren.

Afbeelding 10: Verschillende soorten condensatoren

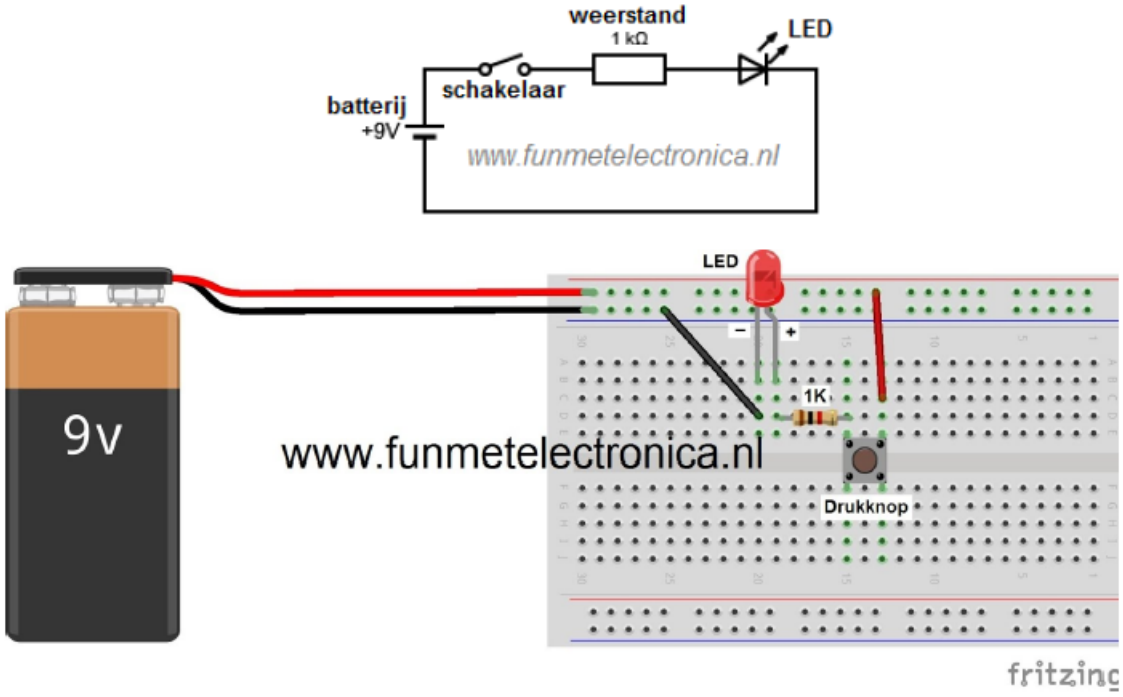
Een lege condensator kan elektrische energie opnemen (elektronen) totdat deze vol is. Een volle condensator kan de elektronen weer loslaten. Hoeveel elektronen een condensator kan opnemen is afhankelijk van de capaciteit. De capaciteit van een condensator wordt weergegeven in farad (F). Omdat de capaciteit van de meeste condensatoren beperkt is gebruikt men vaak microFarad (μF) als eenheid. 1 μF is 0.0000001 F. Elektrolytische condensatoren moeten aan de juiste pool (plus/min) aangesloten worden. In de projecten gebruiken we condensatoren als een vertraging in het circuit. Een stroom loopt totdat de condensator vol is (of leeg), of de spanning bouwt langzaam op of neemt langzaam af.

1.3 Oefeningen

1.3.1 Opdracht 1: led met schakelaar

Opdracht 1.1. Bouw de stroomkring volgens Afbeelding 11, let op de plus- en minkant van de led. Druk de drukknop in en de led brandt. Als de drukknop wordt ingedrukt, kunnen er elektronen door stromen en zal de led branden. Welke richting stromen de elektronen? Van plus naar min of van min naar plus?

Opdracht 1.2. Draai de led om. Wat gebeurt er dan als je op de knop drukt?

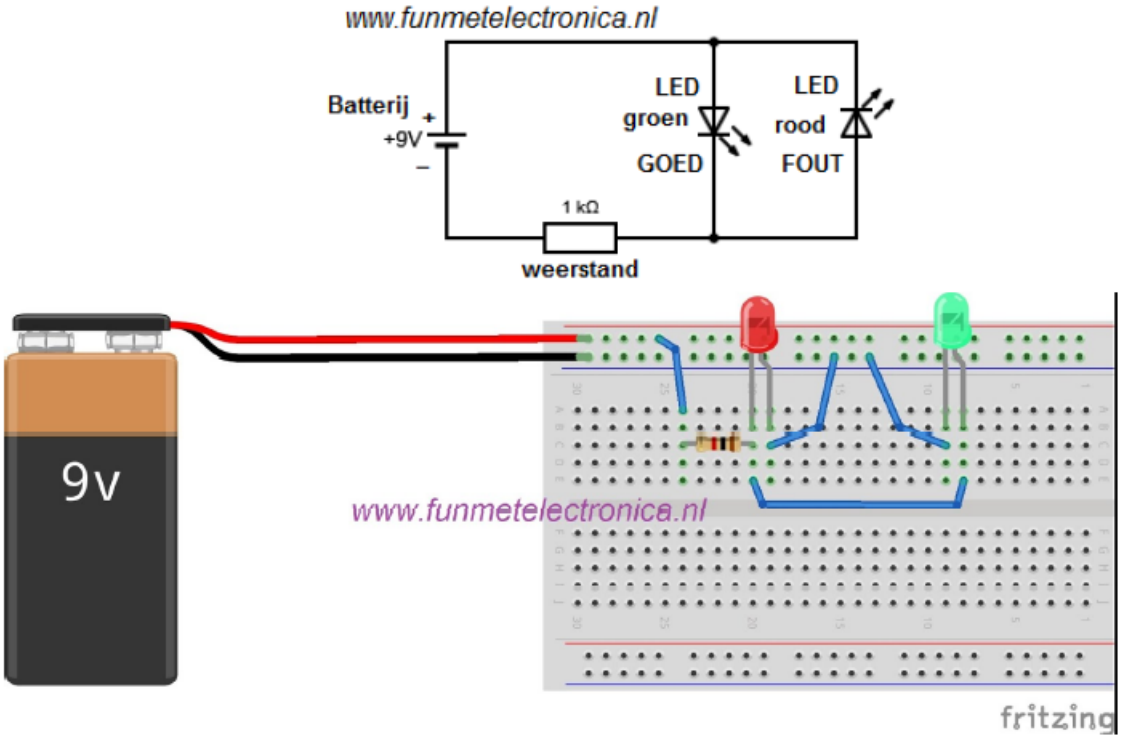


Afbeelding 11: Bouwtekening 1

1.3.2 Opdracht 2: Aansluittester van een batterij

Opdracht 2.1. Bouw de stroomkring volgens Afbeelding 12. Welke kleur led brandt?

Opdracht 2.2. Draai de batterij om. Welke kleur led brandt nu?



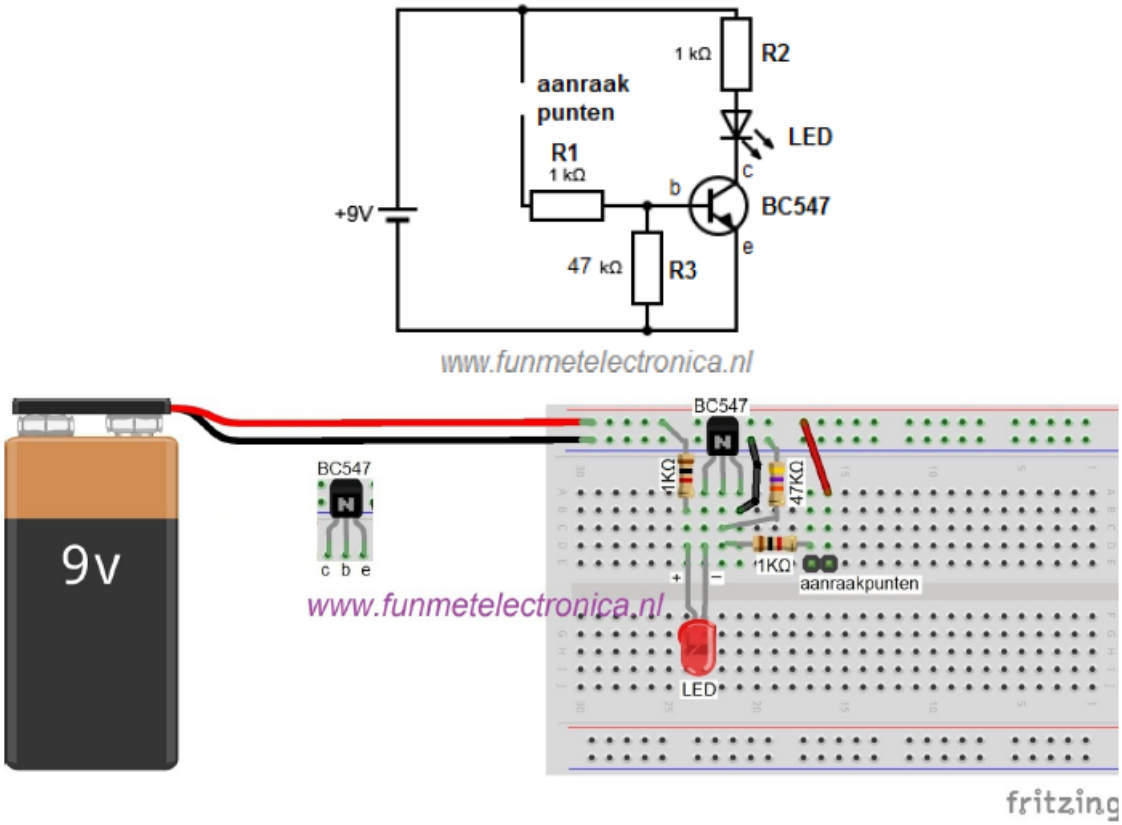
Afbeelding 12: Bouwtekening 2

1.3.3 Opdracht 3: Tiptoets schakeling

Opdracht 3.1. In dit project gebruik je niet een drukknop om een led te schakelen, maar de schakeling word verzorgd door een transistor. Bouw de stroomkring volgens Afbeelding 13. Er gaat een hele kleine stroom via je vinger lopen, deze kleine stroom is voor de transistor genoeg om de led te schakelen.

Opmerking: de weerstand van 47 kΩ mag worden vervangen door een weerstand van 51 kΩ. Waarom gaat de lamp branden als de vinger bij het aanraakpunt in contact komt?

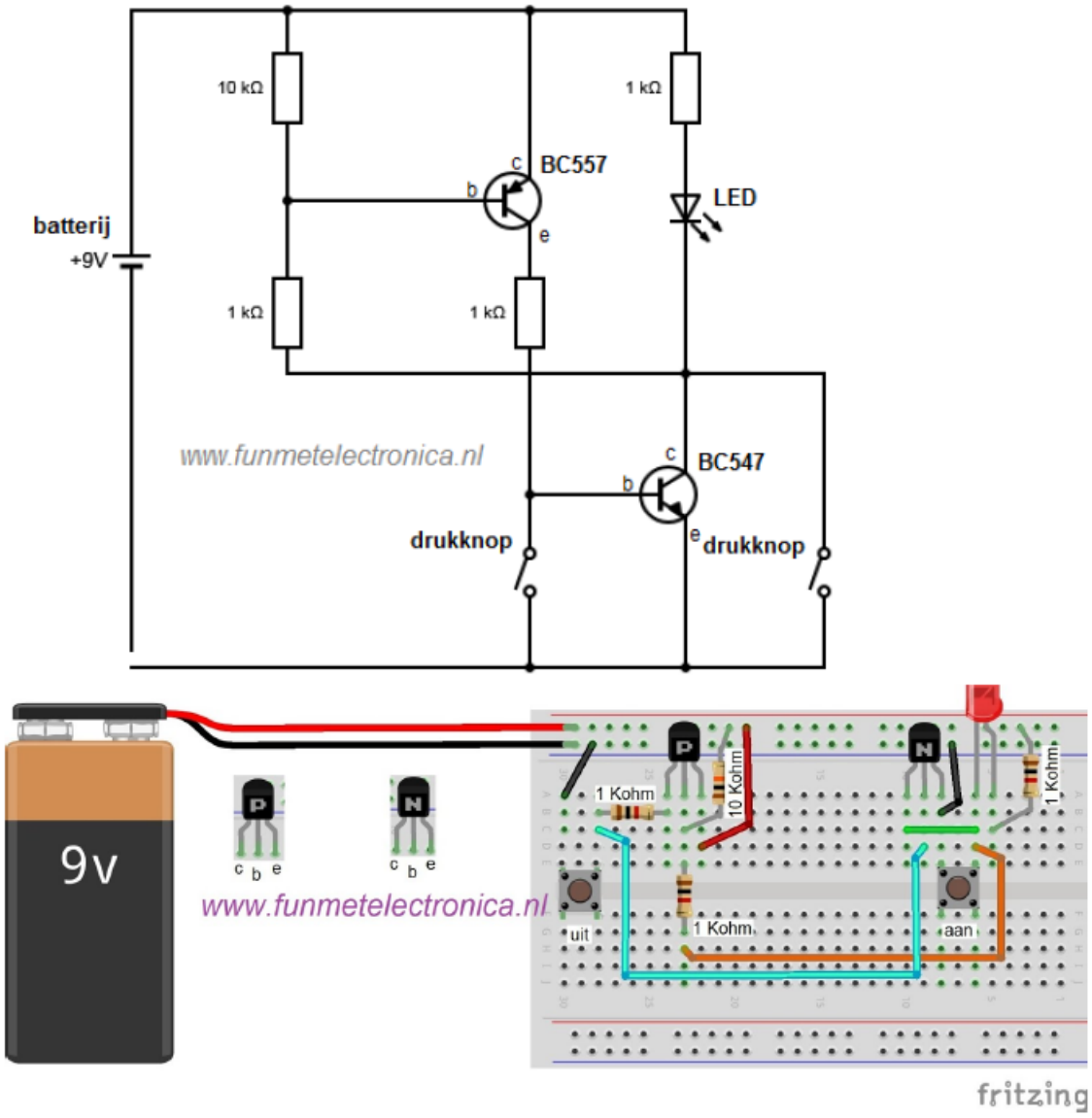
Opdracht 3.2. Maak je vinger nat en druk dan op de contactpunten. Brandt de led nu even fel?



Afbeelding 13: Bouwtekening 3

1.3.4 Opdracht 4: Aan-uit schakeling met aparte drukknoppen

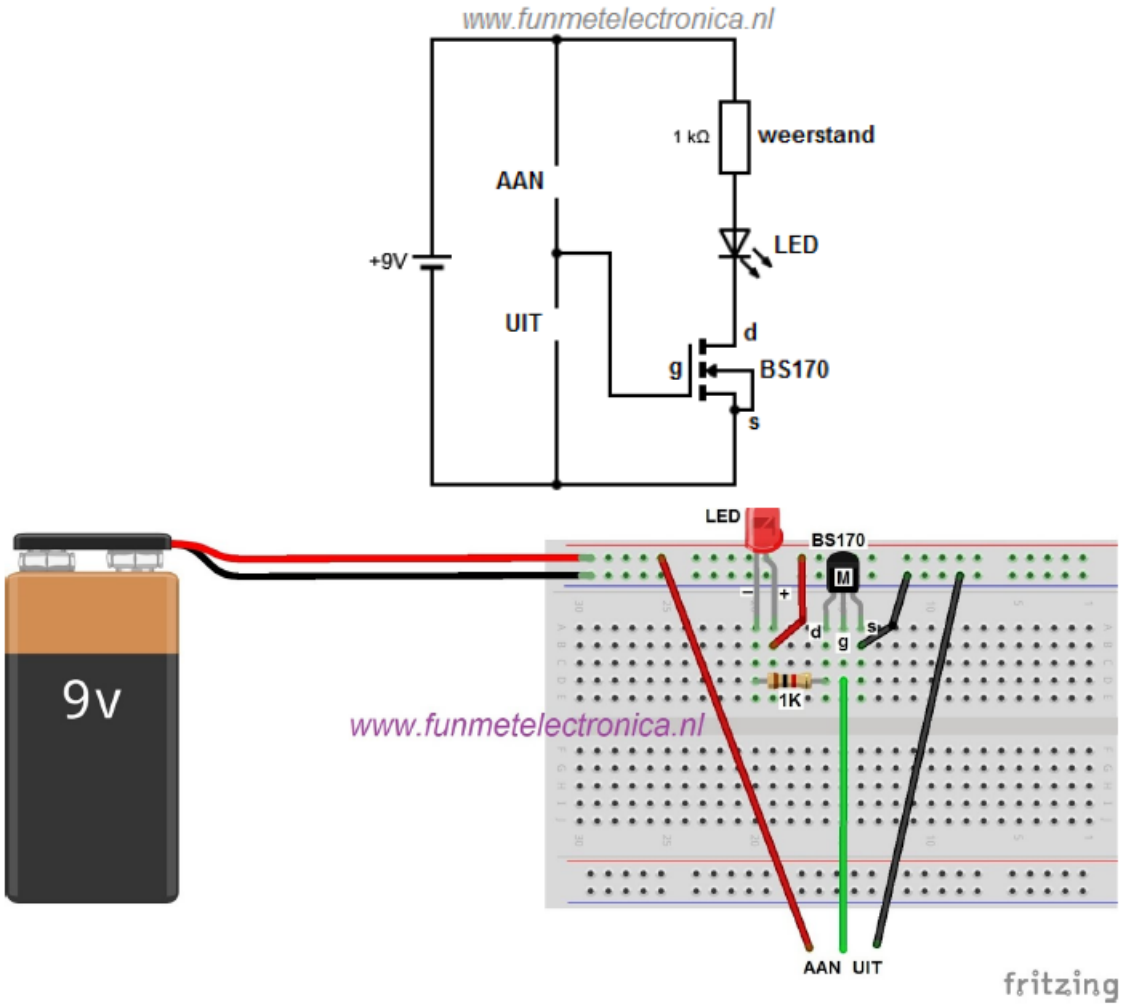
Opdracht 4.1. Bouw de stroomkring volgens Afbeelding 14. De 'aan' knop zal de led aan zetten, ook als je de knop loslaat blijft de led aan. De led gaat pas uit als je op 'uit' knop drukt, waarom is dit?



Afbeelding 14: Bouwtekening 4

1.3.5 Opdracht 5: Aan-uit schakeling met aparte tiptoetsen

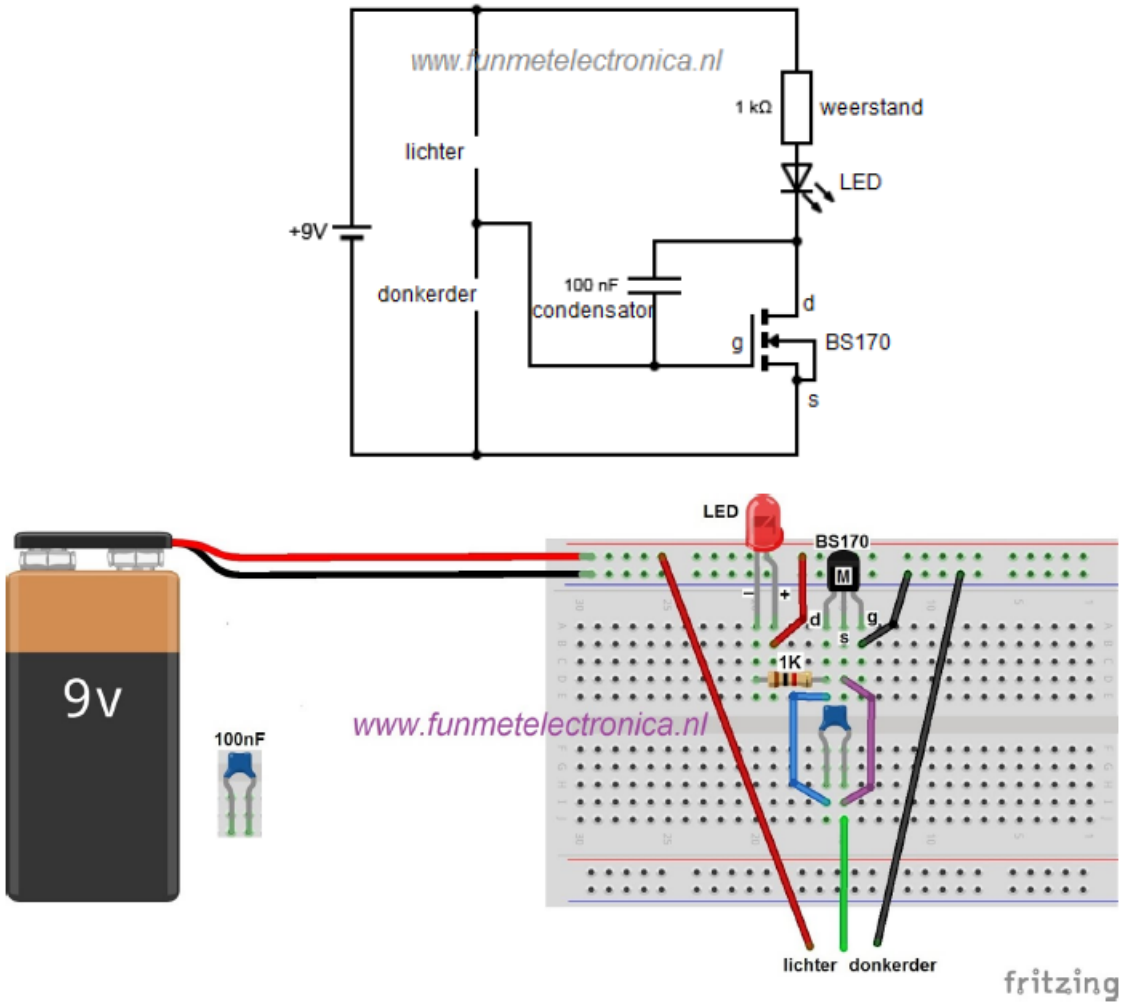
Opdracht 5.1. Dit project lijkt op opdracht 4 alleen wordt hier de schakeling in Afbeelding 15 gemaakt met aanraakpunten. Door het gebruik van de BS170 transistor is de schakeling behoorlijk vereenvoudigd. Wat zou het voordeel en nadelen kunnen zijn ten opzichte van opdracht 4?



Afbeelding 15: Bouwtekening 5

1.3.6 Opdracht 6: Tiptoets dimmer

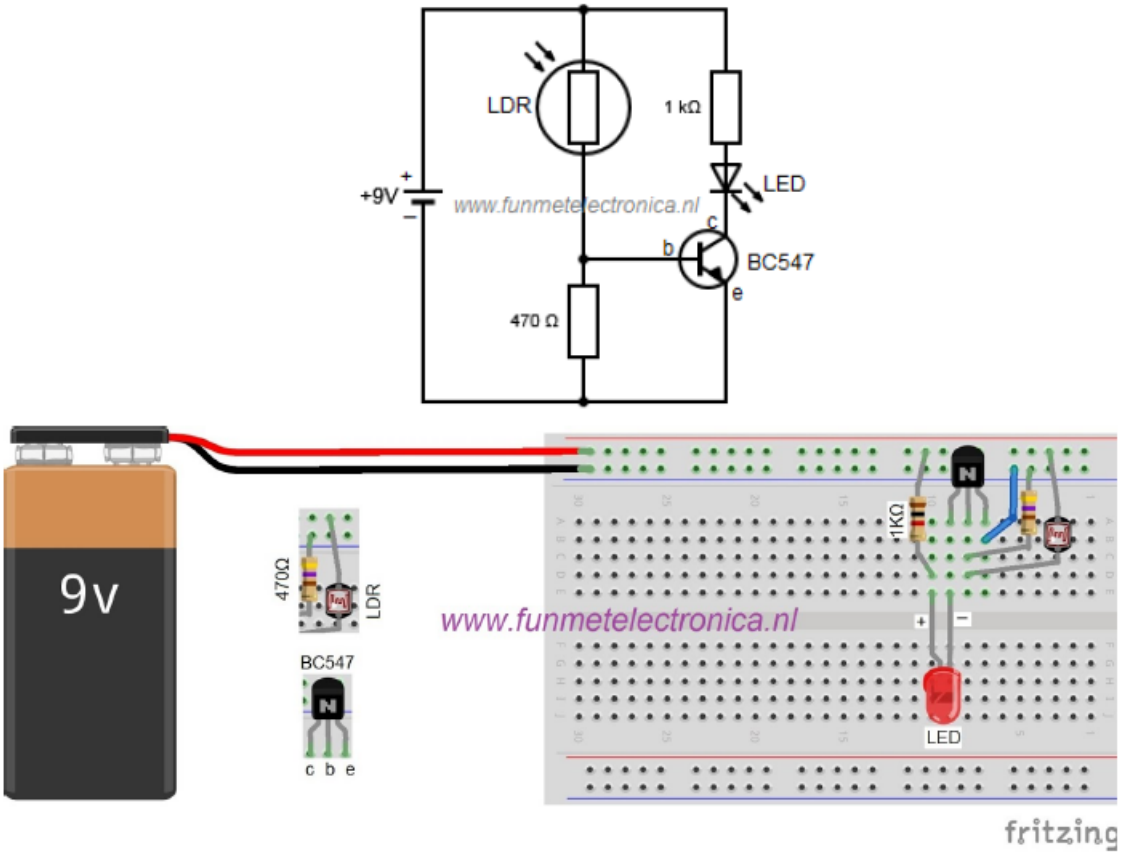
Opdracht 6.1. Bouw de stroomkring volgens Afbeelding 16. Gaat de led bij aanraking niet branden? Probeer dan eens je vinger nat te maken. Let op: 100 nF = 0.1 μF. Zet er eens een condensator extra bij.



Afbeelding 16: Bouwtekening 6

1.3.7 Opdracht 7: Lichtdetector

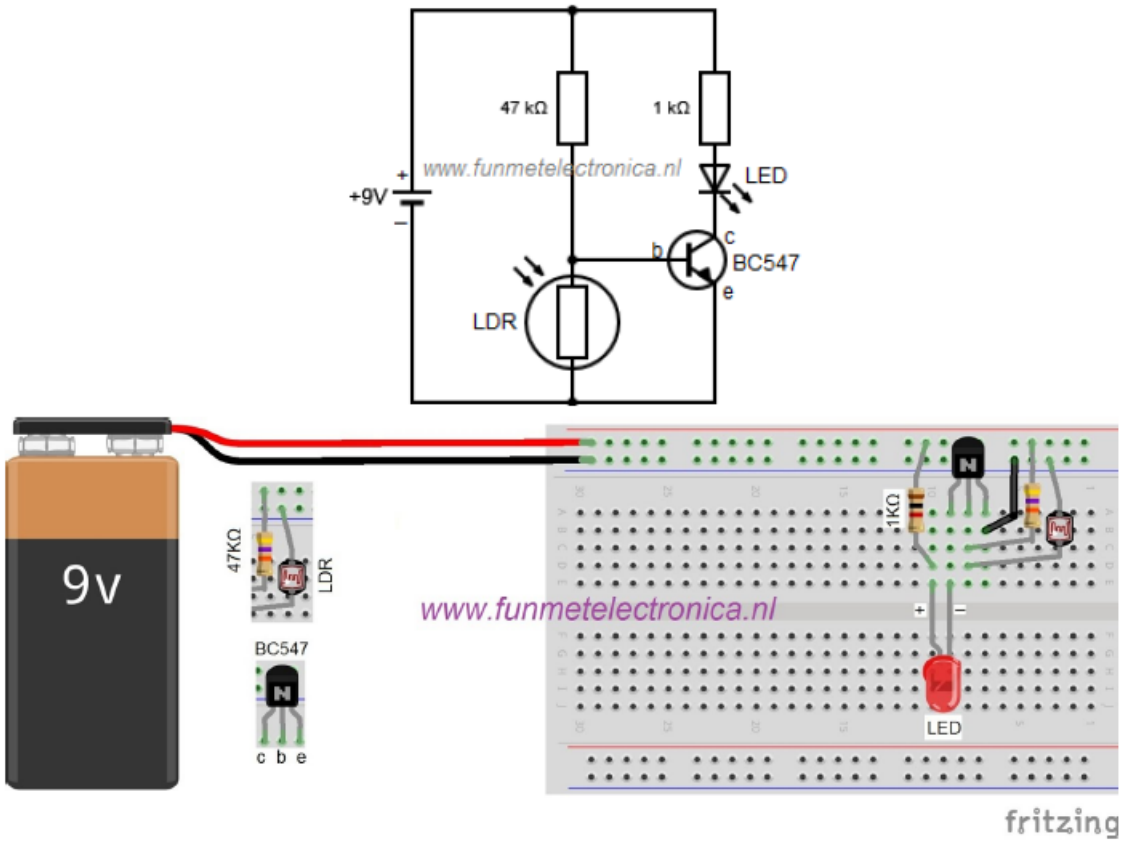
Opdracht 7.1. De led gaat branden als er licht op de sensor (LDR) valt. Bouw de stroomkring volgens Afbeelding 17. Door de weerstand van 470 Ω te veranderen stel je de gevoeligheid van de LDR in. Hoe komt dit?



Afbeelding 17: Bouwtekening 7

1.3.8 Opdracht 8: Schemerschakeling

Opdracht 8.1. De led gaat branden als het donker wordt op de sensor (LDR). Bouw de schakeling volgens Afbeelding 18. Door de weerstand van 47 kΩ te veranderen stel je de gevoeligheid van de LDR in.

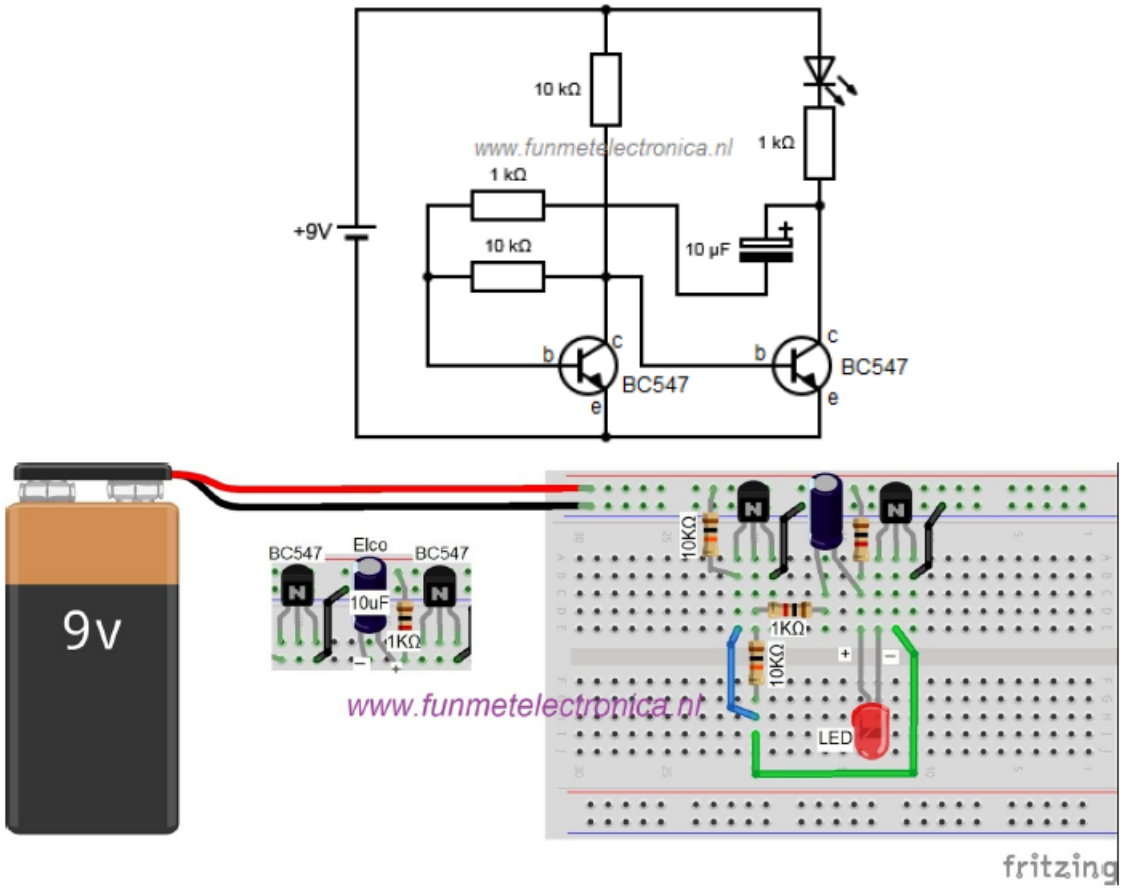


Afbeelding 18: Bouwtekening 8

1.3.9 Opdracht 9: Knipperende led

Opdracht 9.1. Met de schakeling in Afbeelding 19 laat je een led knipperen. Bouw deze opstelling.

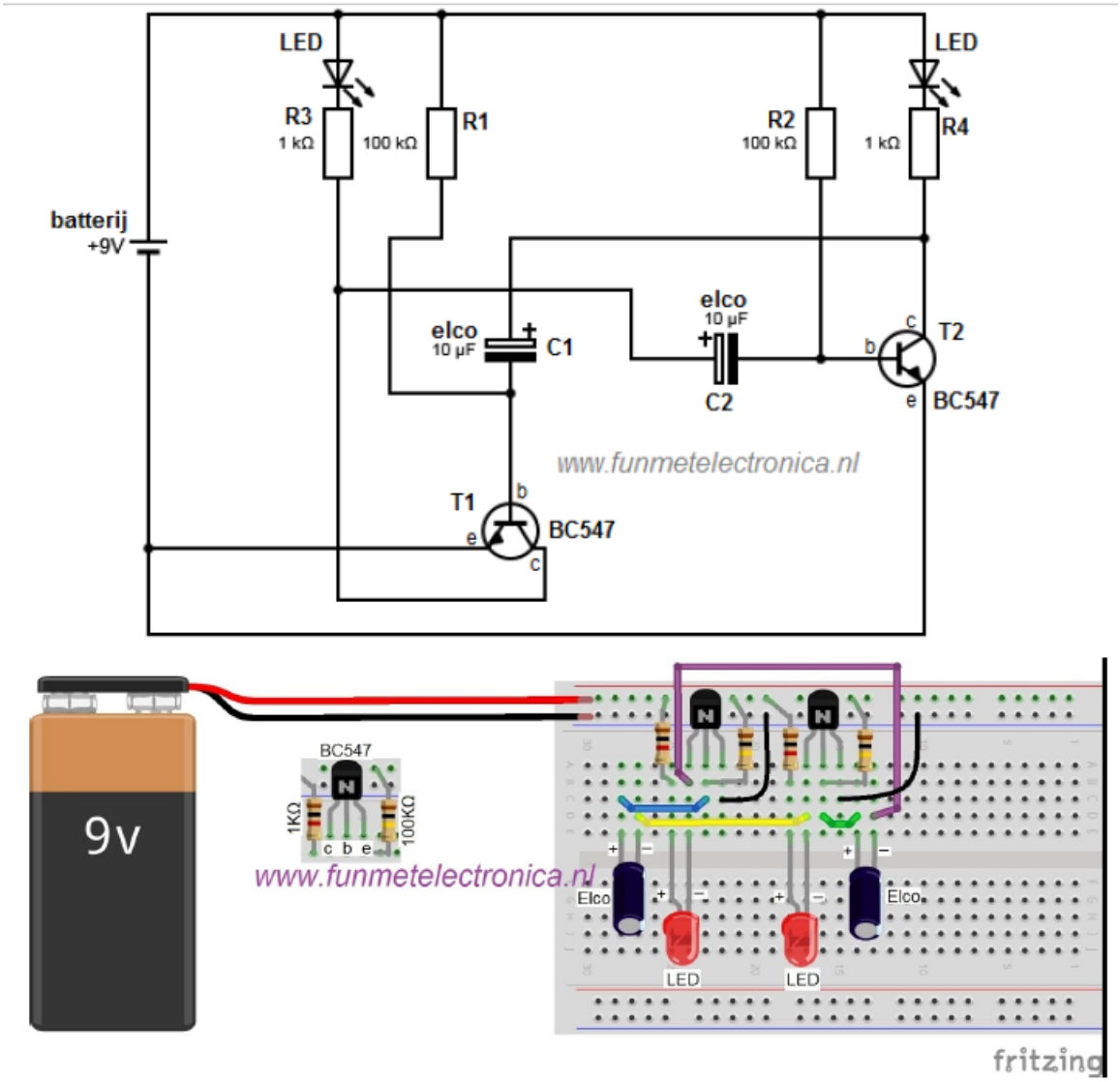
Opdracht 9.2. Vervang de waarde van de condensator 10 μF met 100 μF, kijk wat er gebeurt.



Afbeelding 19: Bouwtekening 9

1.3.10 Opdracht 10: Om en om knipperende leds

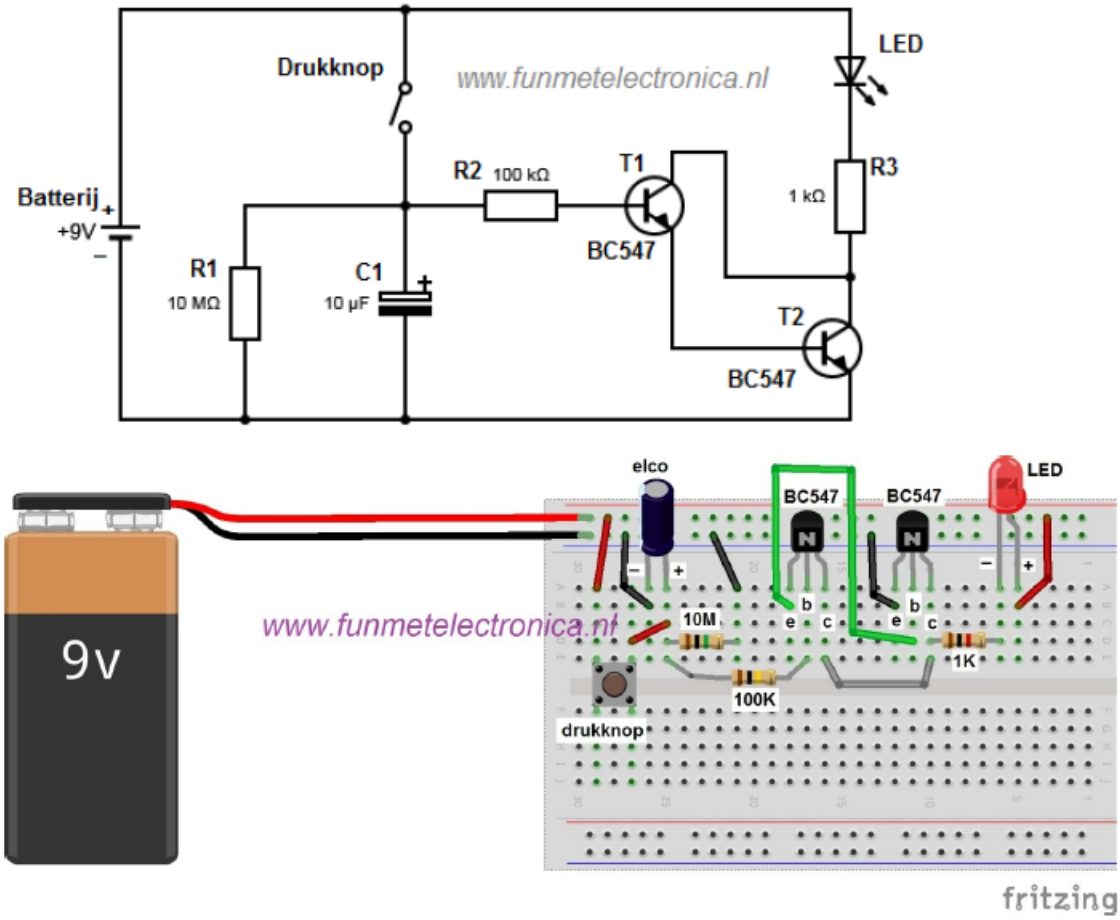
Opdracht 10.1. Met de schakeling gaan de leds om en om knipperen. Bouw de schakeling volgens Afbeelding 20. Verander je de weerstand van 100 kΩ dan zullen de leds sneller of langzamer gaan knipperen.



Afbeelding 20: Bouwtekening 10

1.3.11 Opdracht 11: Timerschakeling

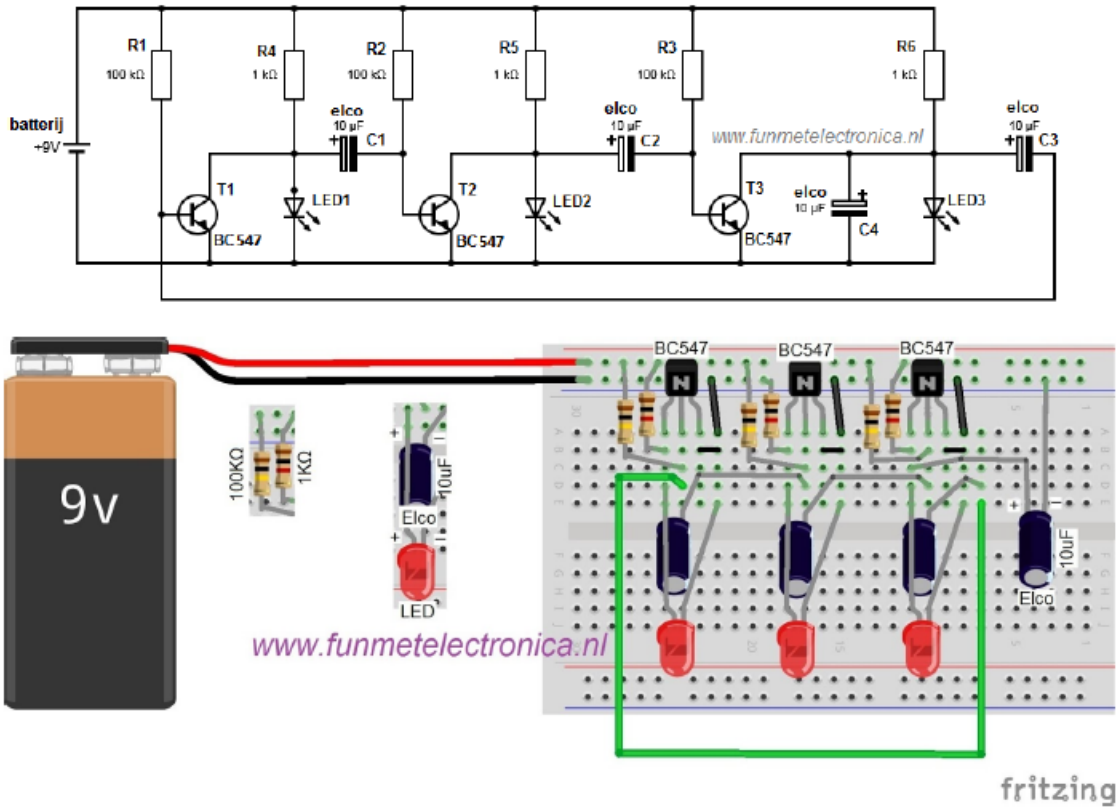
Opdracht 11.1. Door kort op de schakelaar te drukken gaat de led branden, na een tijdje gaat de led langzaam uit. Bouw de stroomkring volgens Afbeelding 21. De tijd is te beïnvloeden door de weerstand R1 te veranderen. Kleinere weerstanden geven een kortere tijd, of door de capaciteit van de condensator aan te passen. Een grotere capaciteit geeft een verlenging van de tijd. Waarom is dit?



Afbeelding 21: Bouwtekening 11

1.3.12 Opdracht 12: Looplicht met 3 leds

Opdracht 12.1. Met deze schakeling maak je een looplicht. Maak de schakeling volgens Afbeelding 22, zoom desnoods een beetje in want deze is pittig. Experimenteer met de waarde van 100 KΩ weerstanden of de waarde van de condensatoren.



Afbeelding 22: Bouwtekening 12

2 Inleiding Programmeren

De software waarin we programmeren is de Arduino IDE. Deze software is te downloaden via `www.arduino.cc` . De programmeertaal die hier gebruikt wordt, is gebaseerd op C en C++. Bij het programmeren moeten we ons aan vaste regels houden, anders begrijpt de computer je niet meer. Soms is het weglaten van een accolade al voldoende om het programma niet te laten werken. Een programma van Arduino noemen we een Sketch. Het programma bestaat uit minstens twee blokken: De `void setup()` en de `void loop()` te zien hieronder.

Code 1: setup.ino

```
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

In deze inleiding zullen de meest belangrijke termen aan bod komen, te beginnen met de `void setup()` en `void loop()`.

2.1 Void loop en setup

2.1.1 `void setup()`

In het blok `void setup()` staat tussen de accolades de informatie. Hier staat wat we allemaal gaan gebruiken. De setup wordt maar één keer ingelezen en vervolgens onthouden.

Code 2: setup.ino

```
1 void setup() {
2   // Tussen de accolades wordt de code 1 keer uitgevoerd.
3   // Dit kan handig zijn om bijvoorbeeld te zeggen waar een led aangesloten
        is.
4 }
5 // Alles wat buiten de accolades staat is niet geldig en zal een ERROR
    geven.
```

2.1.2 `void loop()`

In het blok `void loop()` staat de informatie ook tussen de accolades . In deze informatie staat wat we allemaal gaan doen. De `void loop()` wordt niet 1 keer ingelezen, maar iedere keer herhaald. Binnen het blok `void loop()` kun je kleinere blokjes maken, die staan dan ook weer binnen twee accolades. Het aantal accolades is dus altijd even. Bij het schrijven van een programma kun je natuurlijk wel eens een accolade vergeten. Als je in de sketch naast een accolade gaat staan laat het programma zien waar de tweede staat, dit ter controle.

Code 3: loop.ino

```
1 void loop(){
2 // Alle code tussen de accolades in de loop worden eeuwig herhaald.
3 // Dit kan bijvoorbeeld handig zijn om een lampje te laten knipperen.
4 }
5 // Alle code buiten de accolades zijn niet geldig en geven een ERROR
```

2.2 Speciale Tekens

Arduino gebruikt een aantal verplichte speciale tekens:

- `int i;` Puntkomma. Iedere programmaregel moet worden afgesloten met een puntkomma. Een vergeten puntkomma zorgt ervoor dat een programma niet werkt.
- `void loop(){}` Accolades. Een accolade geeft het begin en einde van een blok informatie weer. Binnen een blok accolades kunnen meerdere kleine blokken met eigen accolades voorkomen.
- `//commentaar` Regel commentaar. In elke regel kan je // plaatsen. De regel achter de // wordt niet meer gelezen door Arduino. Handig voor uitleg/commentaar. Maar het kan ook gebruikt worden om een programmaregel (tijdelijk) uit te schakelen.
- `/*commentaar */` Commentaarblok. Als je uitleg of commentaar over meerdere regels wilt schrijven kan je gebruik maken van de tekens voor commentaar blok. Uitleg over het programma is, zeker voor anderen, heel handig.

Code 4: signs.ino

```
1  // de ; wordt gebruikt na elke regel zodat het programma weet dat er een
       nieuw stukje code komt hierna
2
3 void loop(){ // openings accolade.
4  // dit is een regel commentaar, alleen op 1 regel kan er iets geschreven
       worden
5 int a = 5;
6  /*
7    Met deze tekens kunnen
8    meerdere regels commentaar geschreven worden zonder probleem.
9  */
10 } // sluitings accolade.
11 // accolades geven een blok van code aan, alles tussen de code zal
       uitgevoerd worden.
```

2.3 Variabelen

Variabelen kan je gebruiken om getallen op te slaan in het geheugen om later weer te kunnen gebruiken. Er zijn verschillende types variabelen die meer of minder geheugenplek innemen, afhankelijk van het aantal getallen dat ze moeten opslaan. Hieronder de meest gebruikte variabelen:

1. `int a;` Integer. Het meest gebruikte data type. Heeft geen decimalen en kan een getal opslaan tussen -32.768 en 32.767
2. `boolean a;` Boolean. Kan twee waardes hebben true of false (waar of niet waar)
3. `long a;` Long. Wordt gebruikt als een integer niet groot genoeg is. Long kan getallen opslaan tussen -2.147.483.648 en 2.147.483.647.
4. `float a;` Float. Floating point betekent decimaal. Een floating point kan een decimaal getal opslaan, kost veel geheugenruimte.
5. `char a;` Char. Slaat een toetsenbordteken op in de ASCII code (bv A=65).
6. `String a;` String. Een string is een groep van characters en kunnen dus woorden of zinnen zijn.

Code 5: variables.ino

```
1  // hieronder voorbeelden van de types
2  int a = 5;
3  int b = -7;
4
5  bool check = true;
6  bool niet_check = false;
7
8  long veel = 2000000000;
9  long weinig = -2000000000;
10
11 float decimaal = 0.245;
12 float geen_decimaal = 3.0;
13
14 char character = 'A';
15 char cijfer_als_character = '1';
16
17 String woord = "Woord";
18 String zin = "Ook zinnen kunnen";
```

2.4 Constanten

Een aantal woorden zijn bij de Arduino gekoppeld aan speciale, constante waardes. Let op de hoofdletters en kleine letters.

HIGH en LOW. Worden gebruikt om een Arduino pin aan of uit te zetten.

Code 6: high_low.ino

```
1 digitalWrite(13, HIGH); // dit betekent: zet 5 volt op pin 13
2 int high = HIGH;  // HIGH is equivalent aan 1
3 int low = LOW;    // LOW is equivalent aan 0
```

INPUT en OUTPUT. Bepalen of een pin een input (spanning meten) of output (spanning geven) is.

Code 7: input_output.ino

```
1 pinMode(13, OUTPUT); // pin 13 is een output.
2 int output = OUTPUT;  // OUTPUT is equivalent aan 1
3 int input = INPUT;    // INPUT is equivalent aan 0
```

true en false. Controleren of iets waar of niet waar is. False (niet waar) is 0. Alle waardes die niet nul zijn beschouwt de Arduino als true (waar). Bij vergelijkingen krijg je als uitput deze boolean waardes.

Code 8: true_false.ino

```
1 if(b == true){
2   // doe iets tussen de accolades als b true als waarde heeft
3 }
```

2.5 Besturingsstructuren

if conditie. Bij deze opdracht wordt gecontroleerd of de conditie die tussen de haakjes staat waar is. Als dat zo is wordt de actie uitgevoerd die tussen de accolades staat. Zo niet, dan wordt de actie overgeslagen.

Code 9: if.ino

```
1 if(val == 1) {
2   digitalWrite(LED,HIGH); // als de waarde val 1 is gaat de LED aan
3 }
```

if else conditie. Hier wordt gekozen tussen twee condities. Als de ene waar is, doe dit anders doe dat.

Code 10: if_else.ino

```
1 if(inputPin == HIGH){
2   //voer actie A uit;
3 }else{
4   //voer actie B uit;
5 }
```

`if else if else` conditie. Je kan het nog verder uitbreiden. Let goed op de haakjes en puntkomma's.

Code 11: if_else_if.ino

```
1 if(inputPin < 500){
2   // als inputPin < 500 -> Voer actie A uit;
3 }else if (inputPin <= 1000){
4   // anders als inputPin <= 1000 -> Voer actie B uit;
5 }else{
6   // anders -> voer actie C uit
7 }
```

`for` loop Laat je een stukje programma een bepaald aantal keren uitvoeren. De for loop heeft 3 delen, de variabelen die elke loop veranderd, wanneer de for loop moet stoppen, en hoe de variabelen moet veranderen elke loop. Deze drie delen staan binnen de haakjes van de for loop.

Code 12: for.ino

```
1 for(int i = 0; i < 10; i ++){
2   /*
3     Voer 10 keer iets uit.
4     i begint bij 0: int i = 0;
5     de for loop blijft doorgaan als i kleiner is dan 10: i < 10;
6     na elke loop wordt i 1 groter: i++;
7   */
8 }
```

`while` loop Lijkt wel wat op de for loop. Doe iets zolang aan een bepaalde voorwaarde voldaan wordt. Wordt er niet aan de voorwaarde voldaan dan wordt het blokje overgeslagen. Kijk uit met deze loop, soms kan je de while loop oneindig uitvoeren als je niet oppast!

Code 13: while.ino

```
1 int var = 0;
2 while(var < 200) {
3   // doe iets 200 keer
4   // De int var = 0 en var++ staan nu ergens anders.
5   var++;
6 }
```

2.6 Rekenen

Bij het programmeren kan je verschillende waardes ook veranderen door middel van verschillende operaties.

1. Optellen, optellen van twee waardes kan gedaan worden met het behulp van een + teken.
2. Aftrekken, aftrekken van twee waardes kan gedaan worden met het behulp van een − teken.
3. Vermenigvuldigen, vermenigvuldigen van twee waardes kan gedaan worden met het behulp van een * teken.
4. Delen, delen van twee waardes kan gedaan worden met het behulp van een / teken.
5. Moduleren, moduleren van twee waardes kan gedaan worden met het behulp van een % teken.

Code 14: math.ino

```
1 float a = 13;
2 float b = 3;
3
4 c = a + b; // Optellen, c = 16
5 c = a - b; // Aftrekken, c = 10
6 c = a * b; // Vermenigvuldigen, c = 39
7 c = a / b; // Delen, c = 4.33 (wordt standaard geprint met twee decimalen)
8 c = a % b; // Modulo, Geeft het restgetal van een deling: c = 1
```

2.7 Vergelijken

Er kunnen ook verschillende waardes met elkaar vergeleken worden. Dit wordt vaak gedaan in de besturingssystemen die eerder uitgelegd zijn.

1. Is gelijk aan met ==
2. Is niet gelijk aan met !=
3. Is kleiner dan met <
4. Is groter dan met >
5. Is kleiner of gelijk aan met <=
6. Is groter of gelijk aan met >=

Code 15: compare.ino

```
1 x == y  // x is gelijk aan y
2 x != y  // x is niet gelijk aan y
3 x <  y  // x is kleiner dan y
4 x >  y  // x is groter dan y
5 x <= y  // x is kleiner of gelijk aan y
6 x >= y  // x is groter of gelijk aan y
```

2.8 Logische berekeningen

Logische berekeningen (Boolean operators) zijn vergelijkingen waarbij de uitkomst waar of niet waar is. Ze worden vaak gebruikt in if opdrachten om verschillende condities met elkaar te vergelijken:

1. && Logische en (and): if (x>0 && x<5). Waar als beide vergelijkingen waar zijn.
2. || Logische of (or): if (x>0 !! y>0). Waar als een van de twee waar is.
3. ! Logische niet (not): if (!x > 0). Waar als de vergelijking niet waar is.

Code 16: boolean_operators.ino

```
// and operator
c = false && false;  // false
c = true  && false;  // false
c = false && true;   // false
c = true  && true;   // true

// or operator
c = false || false;  // false
c = true  || false;  // true
c = false || true;   // true
c = true  || true;   // true

// not
c = ! true;          // false
c = ! false;         // true
```

2.9 Digitale functies

Arduino heeft speciale functies voor de digitale pinnetjes.

pinMode Stelt een bepaalde pin in als OUTPUT of als INPUT.

Code 17: pinmode.ino

```
pinMode(7, INPUT);  // Maakt van pin 7 een input.
```

digitalWrite Zet een bepaalde pin hoog (HIGH) of laag (LOW).

Code 18: digitalwrite.ino

```
digitalWrite(9, HIGH);  // Zet pin 9 aan.
```

digitalRead Leest of een pin hoog of laag staat.

Code 19: digitalread.ino

```
val = digitalRead(8); // Geeft de waarde HIGH (wel spanning) of LOW (geen
    spanning) van pin 8.
```

2.10 Analoge functies

analogRead Leest het voltage (tussen 0 en 5 volt) dat staat op een analoge input pin (pin 0 t/m 5 zijn analoge pinnen) en geeft daarvoor een waarde tussen 0 en 1023.

Code 20: analogread.ino

```
1 val = analogRead(2); // Meet de spanning op pin2 en geef een waarde (val)
```

analogWrite Een Arduino is een digitaal apparaat en kan alleen wel of geen stroom zetten op een pin. De pinnen 3, 5, 6, 9, 10 en 11 ondersteunen PWM (pulse width modulation). Deze pinnen kunnen heel snel aan en uit gezet worden waardoor ze analoog lijken. De value kan een getal zijn tussen 0 en 255 die omgeschaald wordt naar een spanning tussen 0 en 5 V.

Code 21: analogwrite.ino

```
1 analogWrite(9, 128); // Dim een led op pin 9 met 50%
```

3 Inleiding Arduino

De Arduino lessen zijn hier als .zip bestand te downloaden. Voor de Arduino lessen is ook een docentenhandleiding beschikbaar. Vraag aan via de mail, `info@funmetelectronica.nl`. Ook zullen alle bestanden en tekeningen te vinden zijn op de website als u verder gaat met de tutorials.

3.1 Installatie Arduino IDE

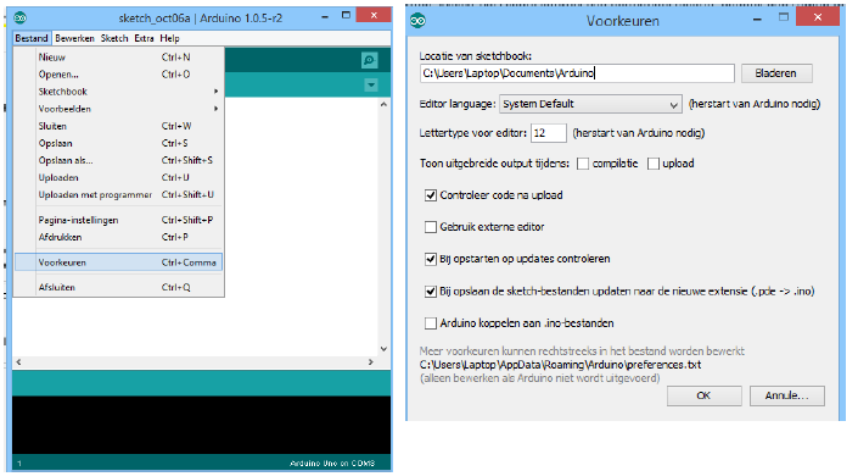
De Arduino IDE is te downloaden via `www.arduino.cc`. Installeer de Arduino IDE op de computer. Bij een standaard installatie maakt Arduino een map aan in mijn documenten voor de lessen. Maakt uw school gebruik van netwerkinstallatie, bedenk dan samen met ICT een oplossing.

Ook is er een USB versie voor Arduino beschikbaar die geen installatie nodig heeft. Deze kan zonder netwerkinstallatie uitgevoerd worden. Een ander probleem die wel kan opkomen is dat de drivers die nodig zijn voor de Arduino niet beschikbaar zijn of niet geïnstalleerd mogen worden. Vraag dan de ICT om een oplossing.

3.2 Installatie ArduBlock

Download Ardublock via `https://sourceforge.net/projects/ardublock/`. Ga na wat de voorkeurs-map van de Arduino IDE is voor sketchbook. Dit doe je door Arduino IDE te openen: ga naar: Bestand, Voorkeuren en vindt in het volgende scherm de locatie van sketchbook. Binnen deze map moeten eerst een drietal mappen worden aangemaakt. In de laatste map plaats je het .jar bestand van ArduBlock. In het specifieke geval van deze laptop van de afbeelding wordt dit:

C:/Users/Laptop/Documents/Arduino/tools/ArduBlockTool/tool/ardublock-all.jar



Afbeelding 23: Voorkeuren

Je moet dus in de map 'Arduino' de map tools aanmaken. 'tools' openen en daarin ArduBlockTool aanmaken. 'ArduBlockTool' openen en daarin tool aanmaken. 'tool' openen en daarin het bestand ardublock-all.jar plaatsen Let op de benaming van de aan te maken mappen, is HOOFDLETTER gevoelig. Sluit Arduino IDE af. Open Arduino IDE opnieuw. Arduino IDE zal nu de ArduBlock plug-in herkennen.

3.3 Gebruik Chromebook

De Arduino lessen van Fun met Electronica zijn ook te gebruiken in de online omgeving van CODEBENDER. Dit is goed nieuws voor Chromebook gebruikers. Ga op een Chromebook naar de Chrome Web Store. En installeer de Codebender App. Sluit je Arduino aan op het Chromebook en vindt de linkjes voor de lessen op de website. *Let op:* Het is nog in de testfase! Heb je vragen of opmerkingen laat het weten via `info@funmetelectronica.nl`.

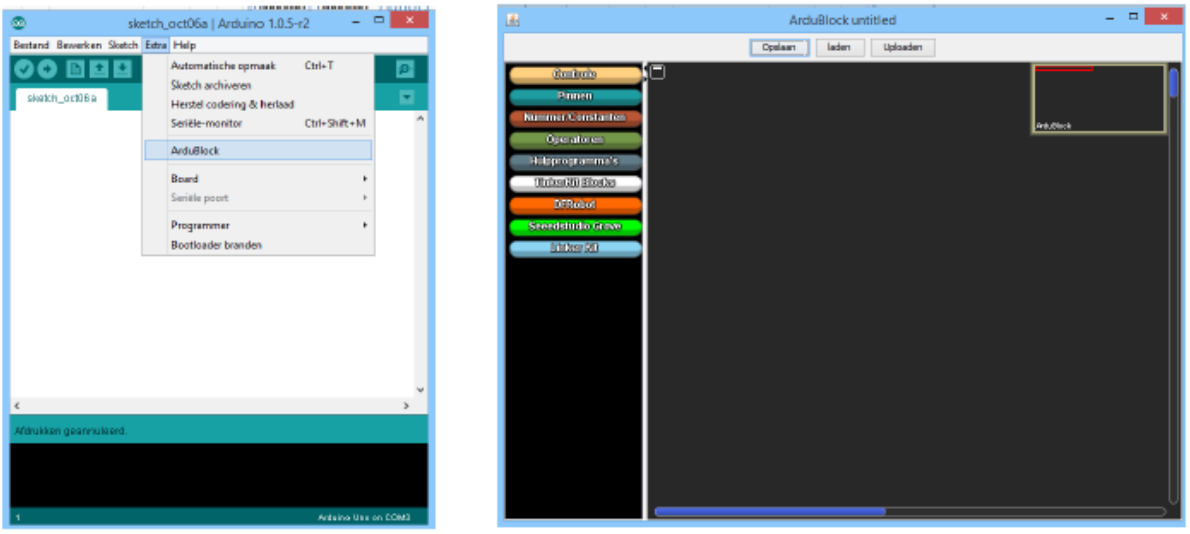
3.4 Gebruik Materiaal en Lessen

3.4.1 Downloaden van het Arduino lessenpakket

Download het lessenpakket via de website. Ga na wat de voorkeurs-map van de Arduino IDE is voor sketchbook. Dit doe je door Arduino IDE te openen: ga naar: Bestand, Voorkeuren en vindt in het voorkeuren-scherm de locatie van sketchbook. Plaats in de 'Arduino' map de Arduino lessen.

3.4.2 Downloaden van het ArduBlock lessenpakket

Download het lessenpakket via de website. Pak het .zip bestand uit en plaats de lessen in de map 'Arduino' die je hierboven ook nodig had om de drie mappen in aan te maken.



Afbeelding 24: ArduBlock

3.4.3 Openen van de lessen

Normaal gesproken kan je via de verkenner een les direct openen. Arduino IDE start vanzelf mee op. Als de lessen niet via de verkenner te openen zijn , start dan eerst de Arduino IDE op en open de les via: Bestand, Open. Voor de ArdubClock lessen volg je deze stappen: Open Arduino IDE. Ga naar: Extra, ArduBlock. ArduBlock opent nu bovenop de Arduino IDE.

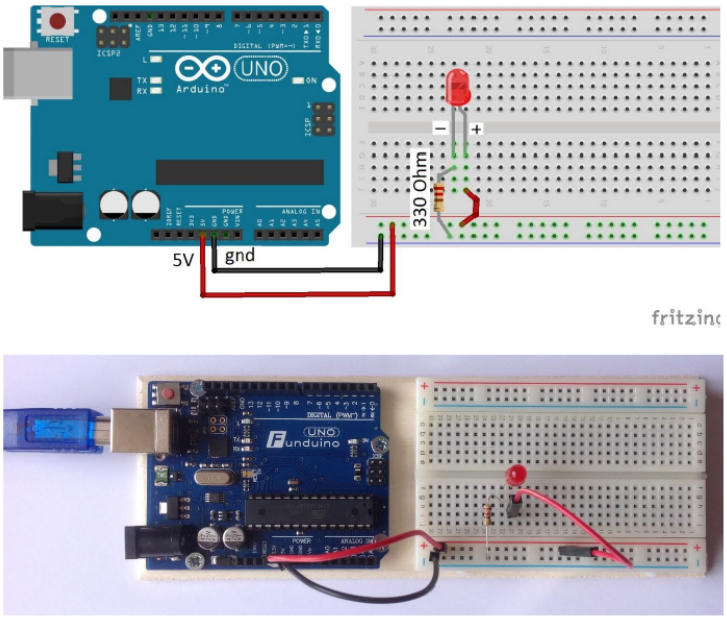
3.4.4 Uploaden van code

Is alles klaar en de Arduino is via USB verbonden? Nu kun je via de knop 'uploaden' in ArduBlock de sketch uploaden. ArduBlock vertaalt nu de grafische blokken naar de Arduino IDE. Gebruik je geen ArduBlock? Druk dan simpelweg op het pijltje in de Arduino IDE.

Na het uploaden zie je de les ook in werking. In Ardublock staat bij elke les een opdracht. Voer deze opdracht in ArduBlock uit en gebruik weer de knop 'uploaden' om het effect van de verandering te zien in Arduino IDE en op je breadboard. Om naar een volgende les te gaan hoef je niets af te sluiten, gebruik in ArduBlock de knop 'laden' om een nieuwe les te kiezen. Sla de gemaakte les eventueel onder een andere naam op. In de Arduino IDE blijft nog wel de oude les staan, maar die verandert automatisch bij het uploaden van elke volgende les.

3.4.5 Gebruik van het materiaal

Voor vele componenten staat de uitleg en meer informatie in de tutorial Basis elektronica/onderdelen. Lees dit eerst goed door. Om de componenten aan elkaar te koppelen moet er uiteindelijk een stroomkring kunnen ontstaan. Maak de onderstaande stroomkring en gebruik de Arduino als voeding. De Arduino moet wel aangesloten zijn op de computer, de Arduino IDE (het programma) heb je nog even niet nodig. Als de LED brandt, is alles goed aangesloten!



Afbeelding 25: Materiaal

3.4.6 Gebruik van de hulptekeningen en teksten

Fun met Electronica heeft bij elke les hulptekeningen en waar nodig ook hulpteksten geplaatst op de tutorial site. Deze zijn eenvoudig via het uitvalscherm te openen. Dezelfde hulptekeningen zijn ook bruikbaar voor de ArduBlock lessen. Voor les 14 en 15 zijn geen hulptekeningen gemaakt. Ik ga er van uit dat je dan uit de lestekst in Arduino IDE weet hoe je de stroomkring moeten opbouwen.

4 Lessen

Fun met Electronica heeft verschillende gratis lessenpakketten voor je samengesteld. Met deze lessen leer je stap voor stap de wereld van electronica of Arduino ontdekken. Vind je programmeren in Arduino IDE te moeilijk? Dan is ArduBlock misschien een oplossing. Om de lessen uit te kunnen voeren heeft Fun met Electronica speciale kits samengesteld. De Arduino en Ardublock lessen zijn te downloaden als .zip bestand. Pak de bestanden uit in de Arduino-map op je computer. Kijk voor meer informatie in de rest van deze tutorial.

Hoe te werk gaan Aan het begin van een les zal een schema gegeven worden. Bouw dit schema eerst na en open de les in Arduino IDE. Als de code geupload is, zou de opstelling moeten werken volgens de beschrijving aan het begin van de les. Is dit niet het geval? Dan ligt het probleem ALTIJD in de opstelling. Misschien zit een jumper wire te ver opzij of is een led verkeerd aagesloten. Als de opstelling het uiteindelijk doet kan je beginnen aan de opdrachten. In deze opdrachten moet je soms de code en/of het schema een beetje aanpassen.

Het aanpassen van code lijkt misschien niet op écht programmeren, maar zo worden bijna alle opstellingen gemaakt. Op het internet wordt een voorbeeld gevonden van de applicatie en de code/schema wordt zodoende aangepast aan de eigen wensen. Er wordt amper zelf nieuwe code geschreven. Dit zou ook raar zijn als iemand anders het grootse gedeelte al gedaan heeft. We gaan dus niet het wiel opnieuw ontwikkelen.

4.1 Les 1: led

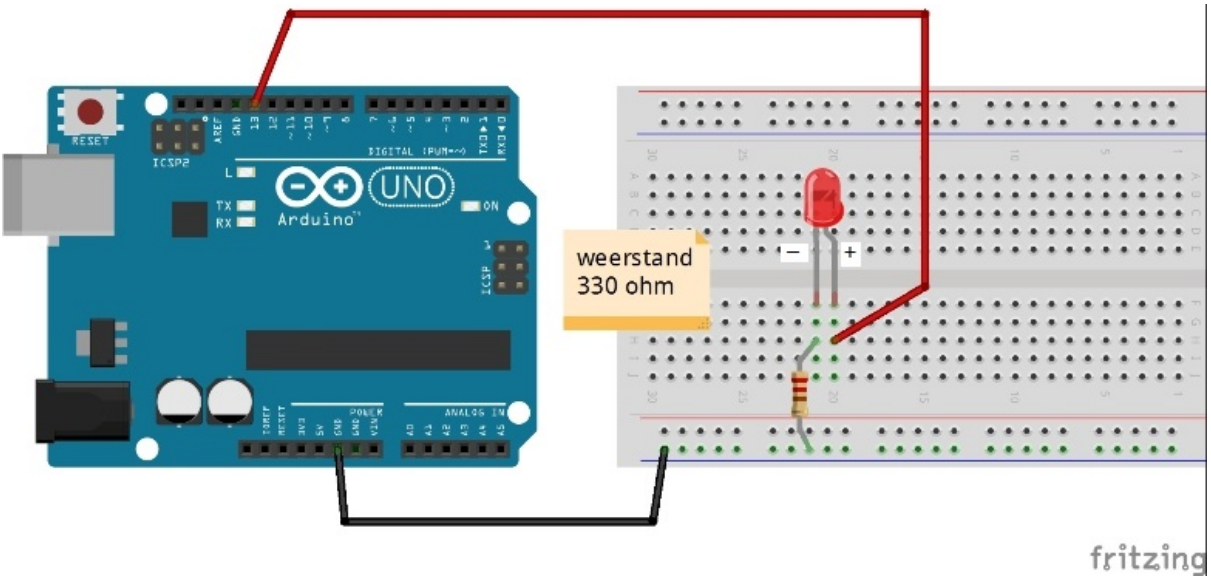
4.1.1 Functie en opdrachten

Als de opstelling correct gebouwd is volgens de bouwtekening en de code geüpload is zal het lampje gaan knipperen.

Opdracht 1.1. Probeer led twee keer zo snel te laten knipperen: verander de tijden.

Opdracht 1.2. Laat de led één keer snel knipperen gevolgd door een langzame knippertijd: voeg een paar stappen toe. Deze onregelmatige snelheid moet zich oneindig herhalen.

4.1.2 Afbeeldingen en Code



Afbeelding 26: Les 1 schema

Code 22: 01_Blink.ino

```
1  int led = 13;                // De led is aangesloten op pin 13
2
3  /*
4    Het programmablok void setup wordt eenmaal beschreven
5  */
6  void setup() {
7    pinMode(led, OUTPUT); // We geven aan dat de led-pin-aansluiting een
         output is (spanning moet geven)
8  }
9
10 /*
11   Het programmablok void loop herhaalt zich keer op keer
12 */
13 void loop() {
14   digitalWrite(led, HIGH);   // zet spanning op de led-pin
15   delay(1000);               // wacht een seconde (1000 miliseconden)
16   digitalWrite(led, LOW);    // zet de spanning van de led-pin af
17   delay(1000);               // wacht een seconde
18 }
```

4.2 Les 2: Meer leds

4.2.1 Functie en opdrachten

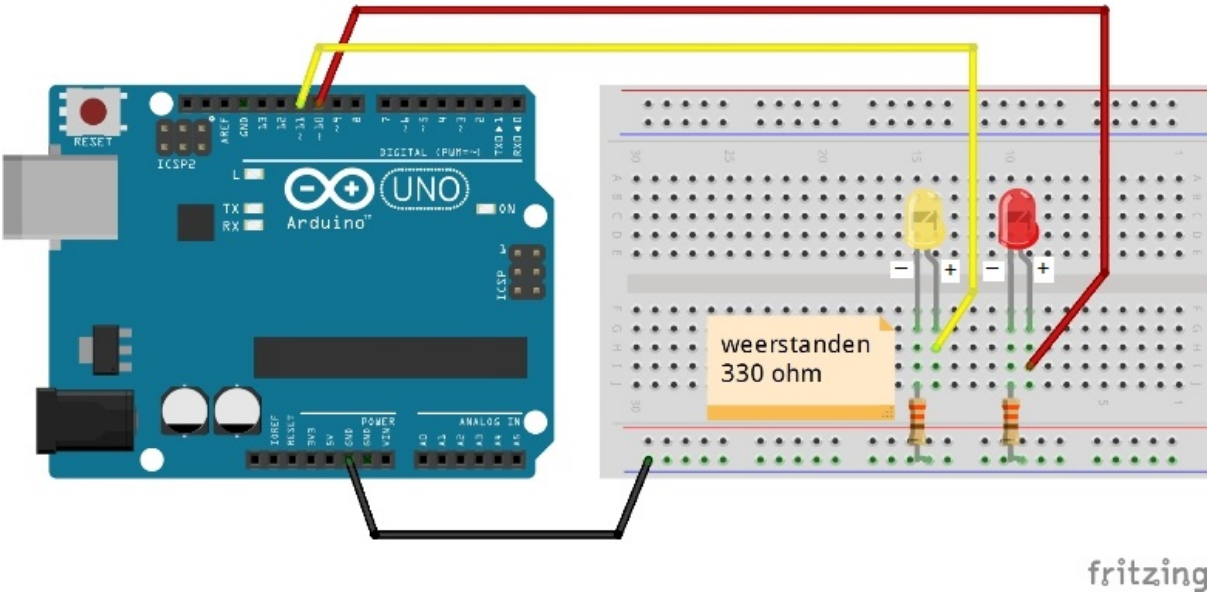
Als de opstelling correct gebouwd is volgens de bouwtekening en de code geüpload is zullen twee lampjes gaan knipperen.

Opdracht 2.1. Zorg dat de lampjes om en om knipperen.

Opdracht 2.2. Voeg een groen lampje toe op pin 12 of 9 en laat deze mee knipperen.

Opdracht 2.3. Maak van deze opstelling een werkende stoplicht met geloofwaardige delays.

4.2.2 Afbeeldingen en Code



Afbeelding 27: Les 2 schema

Code 23: 02_Meerleds.ino

```
1  int ledRood = 10;    // Rode led op pin 10
2  int ledGeel = 11;    // Gele led op pin 11
3
4  void setup() {
5    // We geven aan dat de led-pin-aansluitingen output moet zijn (spanning
       moet geven)
6    pinMode(ledRood, OUTPUT);
7    pinMode(ledGeel, OUTPUT);
8  }
9
10 void loop() {
11   digitalWrite(ledRood, HIGH);  // zet spanning op de led-pin
12   digitalWrite(ledGeel, HIGH);  // zet spanning op de led-pin
13   delay(1000);                  // wacht een seconde (1000 miliseconden)
14   digitalWrite(ledRood, LOW);   // zet de spanning van de led-pin af
15   digitalWrite(ledGeel, LOW);   // zet de spanning van de led-pin af
16   delay(1000);                  // wacht een seconde
17 }
```

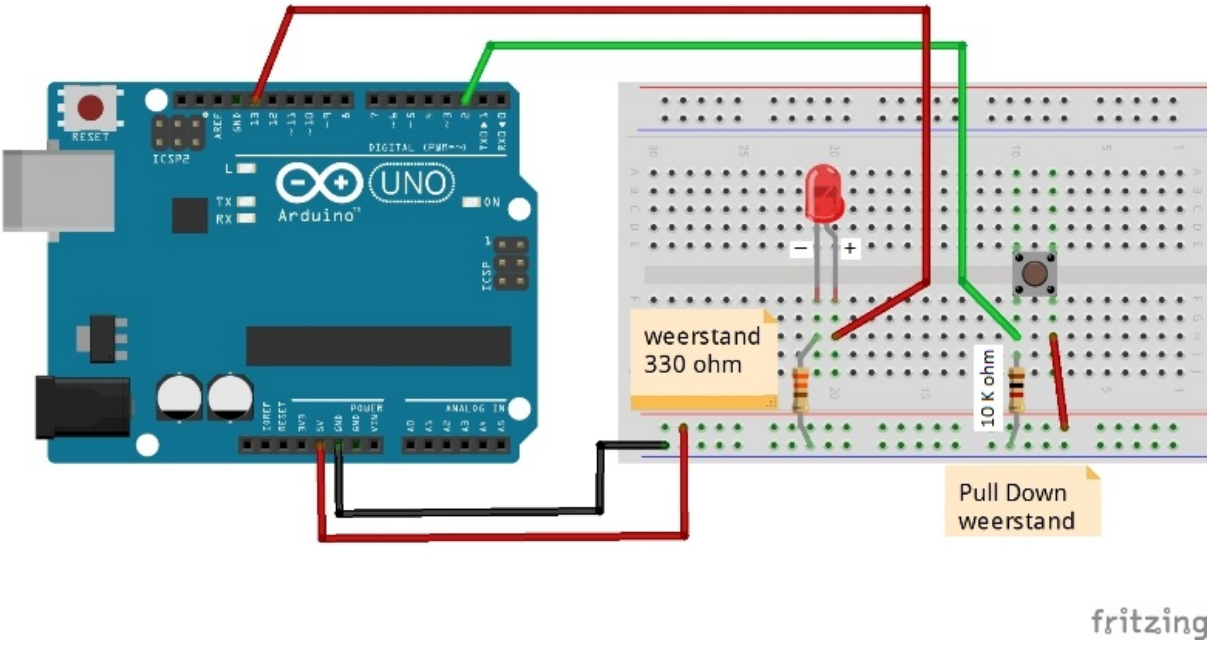
4.3 Les 3: Knop

4.3.1 Functie en opdrachten

Als de opstelling correct gebouwd is volgens de bouwtekening en de code geüpload is zal het lampje branden als er op het knopje gedrukt wordt.

Opdracht 3.1. Verander de sketch zodat het lampje 5 seconden lang blijft branden na het indrukken van de knop. *Tip:* Er zal maar één regel toegevoegd moeten worden in de code.

4.3.2 Afbeeldingen en Code



Afbeelding 28: Les 3 schema

Code 24: 03_Knop.ino

```
1  int knop = 2;           // knop aan pin 2
2  int led =  13;          // led aan pin 13
3  int toestandknop = 0;   // variabele voor het lezen van de knop
4
5  void setup() {
6    pinMode(led, OUTPUT);     // ledpin is output
7    pinMode(knop, INPUT);     // knop is input
8  }
9
10 void loop() {
11   toestandknop = digitalRead(knop); // toestandknop is de waarde van knop
12
13   if (toestandknop == HIGH) {       // controleer of de knop ingedrukt is
14     digitalWrite(led, HIGH);        // indien ingedrukt: led aan
15   } else {
16     digitalWrite(led, LOW);         // anders: led uit
17   }
18 }
```

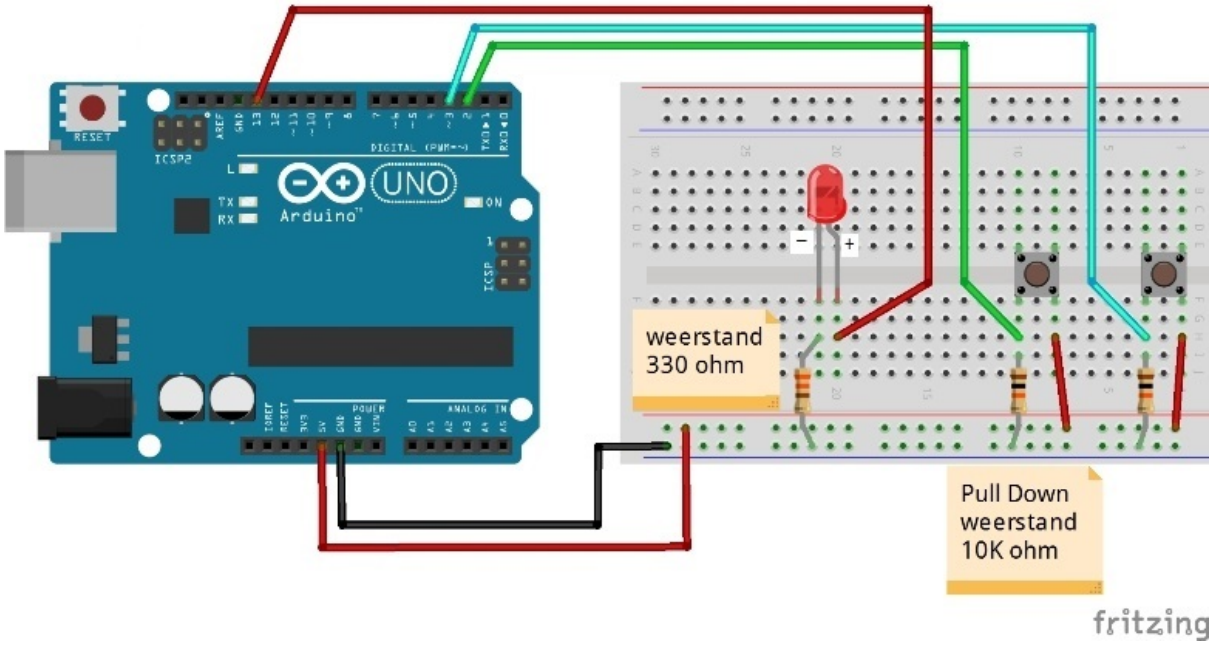
4.4 Les 4: En Of functie

4.4.1 Functie en opdrachten

Als de opstelling correct gebouwd is volgens de bouwtekening en de code geüpload is zal het lampje gaan branden als je één van de twee knopjes ingedrukt hebt. Dit kan handig zijn als je op twee plekken een knopje wilt plaatsen. Dit kan door middel van een 'if' functie met een vergelijking. In de if functie wordt gekeken of knop 1 OF knop 2 ingedrukt is. Dat is mogelijk gemaakt door de tekens || tussen de toestandsknop waardes te zetten. De || tekens werken als een OF.

Opdracht 4.1. Verander de sketch zo dat: De lamp juist aan gaat als de knop 1 EN 2 wordt ingedrukt. Kijk op de tutorial-site bij Introductie Programmeren voor de tekens. Onder het gedeelte 'Logische Berekeningen' is de OF functie te vinden.

4.4.2 Afbeeldingen en Code



Afbeelding 29: Les 4 schema

Code 25: 04_OF_EN.ino

```
1  int knop1 = 2;           // knop aan pin 2
2  int knop2 = 3;           // knop aan pin 3
3  int led =  13;           // led aan pin 13
4  int toestandknop1 = 0;   // variabele voor het lezen van de knop 1
5  int toestandknop2 = 0;   // variabele voor het lezen van de knop 2
6
7  void setup() {
8    pinMode(led, OUTPUT);    // ledpin is output
9    pinMode(knop1, INPUT);   // knop1 is input
10   pinMode(knop2, INPUT);   // knop2 is input
11 }
12
13 void loop() {
14   toestandknop1 = digitalRead(knop1);                    // toestandknop
        is de waarde van knop1
15   toestandknop2 = digitalRead(knop2);                    // toestandknop
        is de waarde van knop2
16
17   if (toestandknop1 == HIGH || toestandknop2 == HIGH) {    // controleer of
        de knop 1 OF 2 ingedrukt is
18     digitalWrite(led, HIGH);                             // indien 1 of 2
          ingedrukt: led aan
19   } else {
20     digitalWrite(led, LOW);                              // anders: led
          uit
21   }
22 }
```

4.5 Les 5: analogSerial

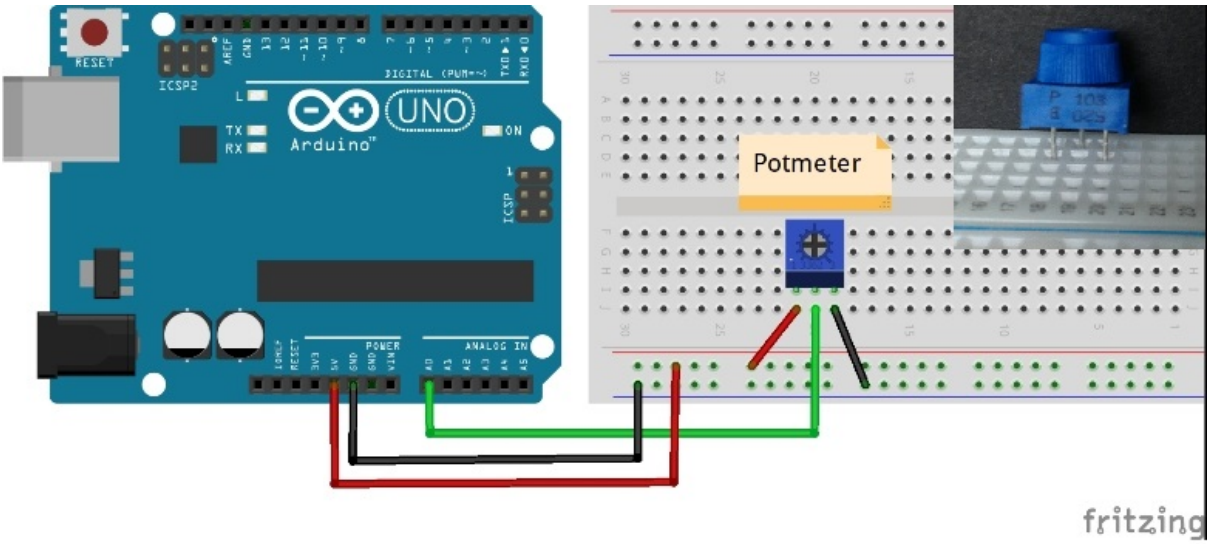
4.5.1 Functie en opdrachten

In deze sketch gaan we meten op een analoge ingang (A0). Arduino kan een spanning meten tussen 0 en 5 Volt. De spanning die gemeten wordt, kan weergegeven worden op de computer. Dit gebeurt via de seriele monitor (rechts boven). De waarde die weer wordt gegeven is tussen 0 en 1023. 0 = 0 volt, 1023 = 5 Volt We gebruiken in het eerste voorbeeld een variabele weerstand (potmeter). De middelste aansluiting van de potmeter is aangesloten op de loper, de spanning varieert omdat op de loper de spanning wordt gedeeld. De spanningsdeler is uitgelegd bij de Ondersteuning op de website.

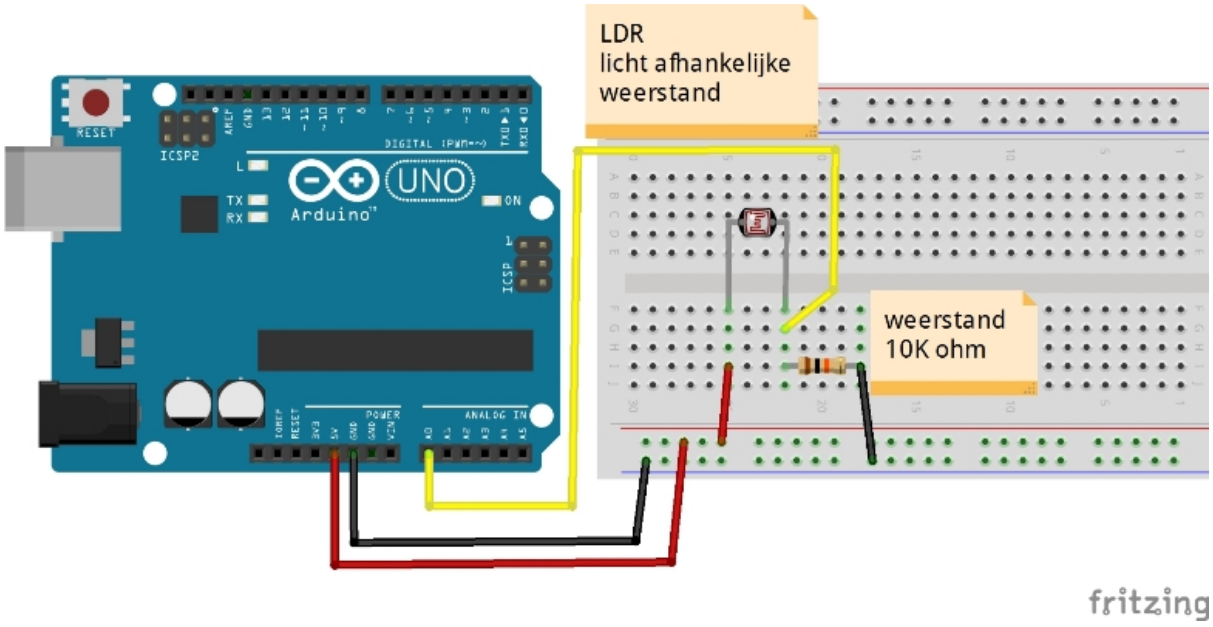
Opdracht 5.1. Lees de waarde van de potmeter uit in de seriele poort. Deze is te vinden rechtsboven in de IDE zoals te zien in het plaatje hieronder.

Opdracht 5.2. Een potmeter aflezen is leuk maar we willen ook met sensoren werken. Sluit een lichtsensor (LDR) aan volgens tekening hieronder. Nu wordt een echte sensor gebruikt. We meten!

4.5.2 Afbeeldingen en Code



Afbeelding 30: Les 5 schema: Potmeter



Afbeelding 31: Les 5 schema: LDR

Code 26: 05_AnalogSerial.ino

```
1 int analogIn = A0;  // Analoge input van de potmeter
2 int potmeter = 0;   // waarde te meten van A0 noemen we potmeter
3
4 void setup() {
5   Serial.begin(9600); // maak een seriele communicatie op 9600 bps
6 }
7
8 void loop() {
9   potmeter = analogRead(analogIn);     // lees de analoge waarde:
10
11  Serial.print("potmeterwaarde = ");   // print "potmeterwaarde = "
12  Serial.println(potmeter);           // print de potmeterwaarde
13
14  delay(50);                          // wacht 50 miliseconden
15 }
```

4.6 Les 6: analogDigital

4.6.1 Functie en opdrachten

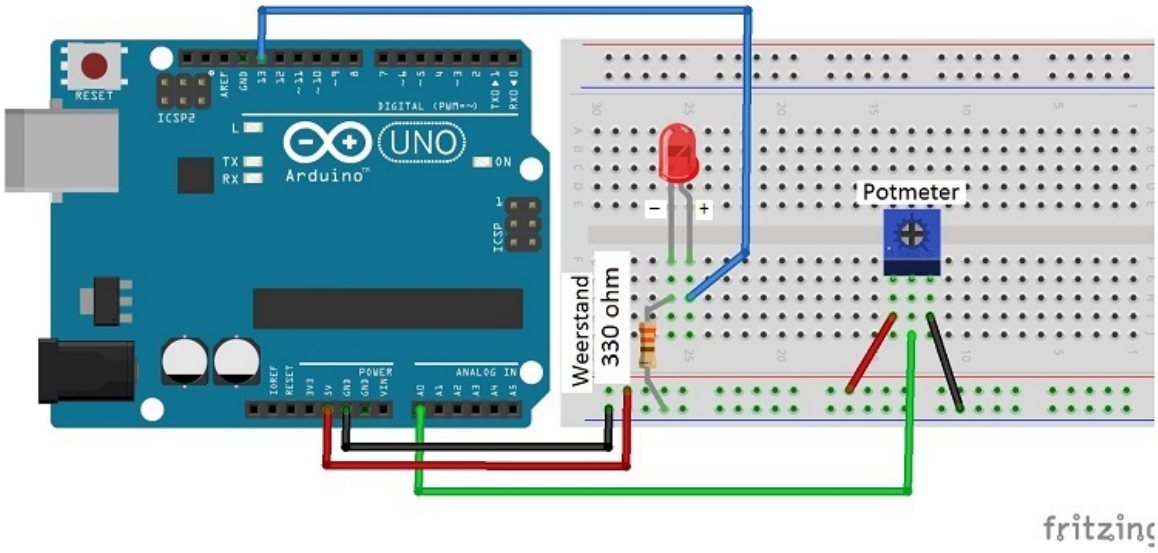
In deze sketch gaan we meten op een analoge ingang (A0). We gaan de waarde van A0 nu gebruiken om een led aan en uit te schakelen. De waarde van A0 stellen we in met een potmeter. We lezen de waarde om te kunnen zien of bij de juiste waarde geschakeld wordt.

Opdracht 6.1. Sluit in plaats van een potmeter een LDR aan (hint: gebruik de tekening van de vorige les). Werkt de schakeling nu al meteen met de LDR of moet je waardes in de code veranderen?

Opdracht 6.2. Verander de code zo (alleen de waardes), dat het lampje gaat branden als er minder licht valt op de LDR. Dus als het donker wordt.

Opdracht 6.3. (Optioneel en moeilijk) Laat de led knipperen uit zichzelf zonder delay door deze op de LDR te richten. Als de led aan gaat zou de waarde van de LDR hoger moeten worden. Gebruik dit om de led te laten knipperen.

4.6.2 Afbeeldingen en Code



Afbeelding 32: Les 6 schema

Code 27: 06_AnalogDigital.ino

```
1  int analogIn = A0;  // Analoge input van de potmeter
2  int potmeter = 0;   // waarde te meten van A0 noemen we potmeter
3  int led = 13;       // de led is op pin 13 aangesloten
4
5  void setup() {
6    pinMode (led, OUTPUT);  // led is een output
7    Serial.begin(9600);     // maak een seriele communicatie op 9600 bps:
8  }
9
10 void loop() {
11   potmeter = analogRead(analogIn);    // lees de analoge waarde:
12
13   Serial.print("potmeterwaarde = ");  // print "potmeterwaarde = "
14   Serial.println(potmeter);           // print de potmeterwaarde
15
16   if (potmeter < 500) {
17     digitalWrite (led, HIGH);         // als de waarde hoger is dan 500
           gaat het lampje aan
18   } else {
19     digitalWrite(led, LOW);           // anders is het lampje uit
20   }
21
22   delay(50);                          // wacht 50 miliseconden
23 }
```

4.7 Les 7: Tone keyboard

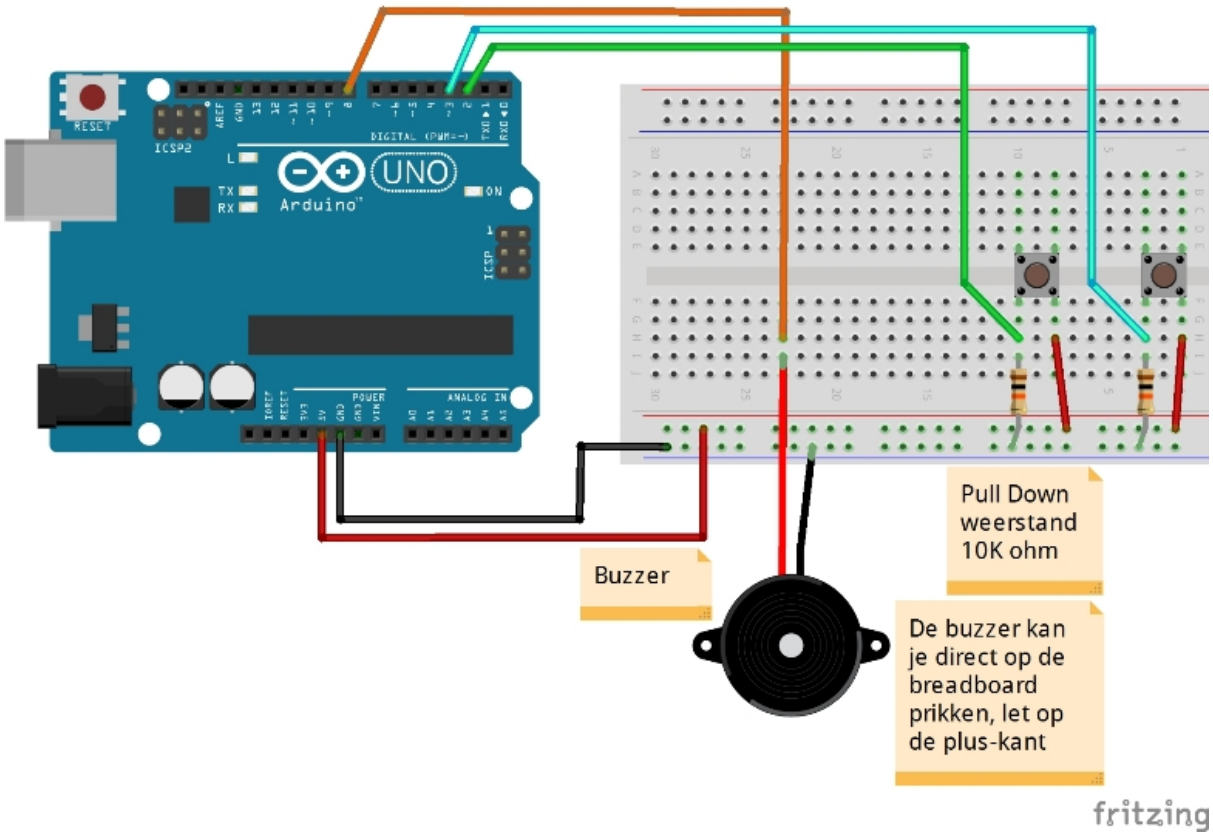
4.7.1 Functie en opdrachten

In deze sketch gaan we de Arduino ombouwen tot een keyboard. We beginnen met een keyboard met 2 toetsen (knoppen). De code gebruikt frequenties uit gangbare tabellen. In de code staan enkele frequenties voor een toon weergegeven. Heb je een speaker i.p.v. een buzzer, sluit je die liever aan.

Opdracht 7.1. Voeg een derde knop toe voor een nieuwe toon.

Opdracht 7.2 (Optioneel). In plaats van knoppen gebruik een LDR. Kan je nu een 'oneindig' aantal tonen gebruiken? Kijk naar de functie `tone(speaker, TONE)`, in het eerste gedeelte van wordt de speakerpin gegeven. In het tweede gedeelte de tone. Dit is een waarde die de frequentie voorsteld. Dit kunnen we vervangen voor de LDR waarde. Als we onze hand dan dichter bij de LDR leggen zou het geluid omlaag moeten gaan. Dit zou een simpele afstandsmeter kunnen zijn.

4.7.2 Afbeeldingen en Code



Afbeelding 33: Les 7 schema: Buzzer

Code 28: 07_ToneKeyboard.ino

```
/*
  Hier worden de noten met bijbehorende frequentie beschreven.
  Kijk maar eens in het tabblad pitches.h,
  op elke regel is het laatste getal de frequentie van de toon.
 */
#include "pitches.h"

const int knop1 = 2;         // knop 1 is aangesloten op pin 2
const int knop2 = 3;         // knop 2 is aangesloten op pin 3

const int speaker = 8;       // de buzzer of speaker is aangesloten op pin 8
const int toon1 = NOTE_C4;   // de toon die hoort bij de noot
const int toon2 = NOTE_D4;   // de toon die hoort bij de noot

int toestandknop1 = 0;       // de waarde van knop 1
int toestandknop2 = 0;       // de waarde van knop 2

void setup() {
  pinMode(knop1, INPUT);       // knop1 is een input
  pinMode(knop2, INPUT);       // knop2 is een input
  pinMode (speaker, OUTPUT);   // speaker is een output
}

void loop() {
  toestandknop1 = digitalRead(knop1); // lees toestand knop1
  toestandknop2 = digitalRead(knop2); // lees toestand knop1

  if (toestandknop1 == HIGH) {
    tone(speaker, toon1);             // toon 1 klinkt bij knop1
  } else if (toestandknop2 == HIGH) {
    tone(speaker, toon2);             // toon 2 klinkt bij knop2
  } else {
    noTone(speaker);                  // geen toon zonder knop ingedrukt
  }
}
```

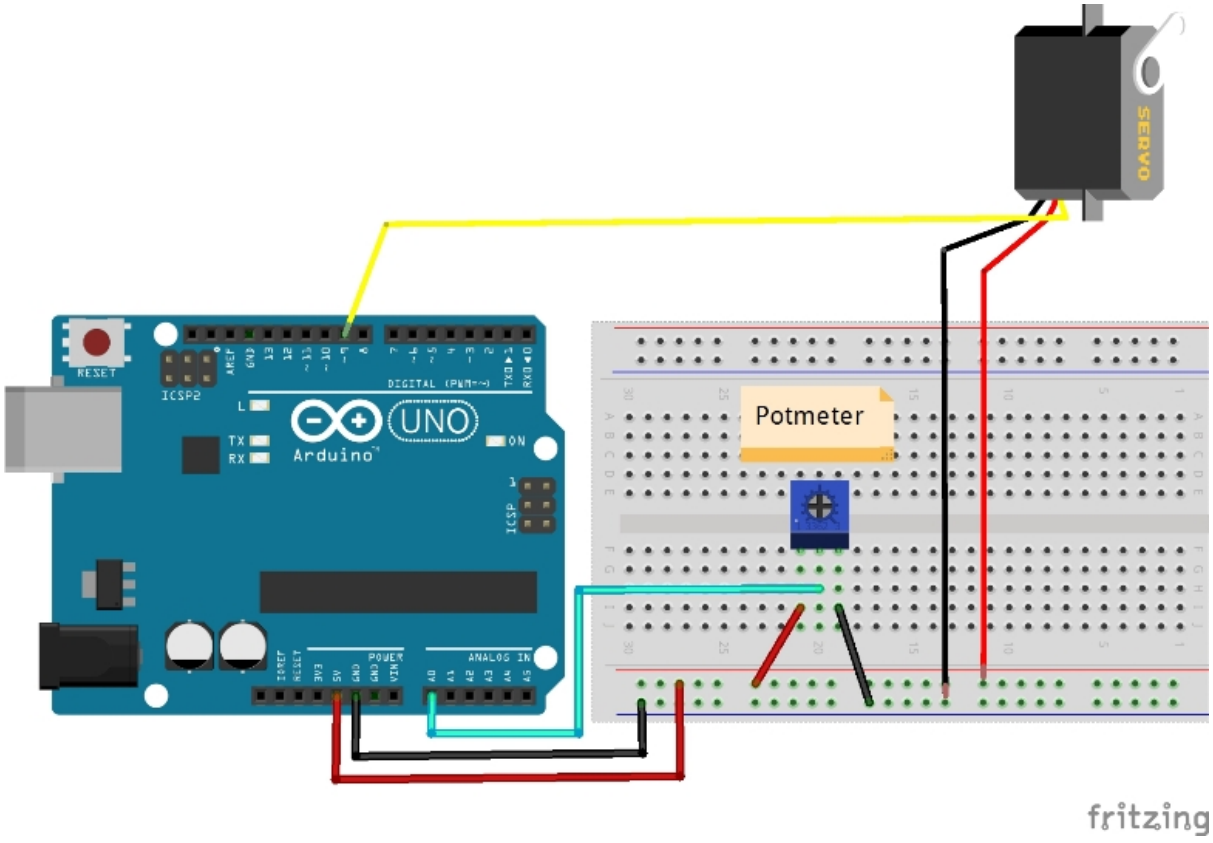
4.8 Les 8: Servo met pot

4.8.1 Functie en opdrachten

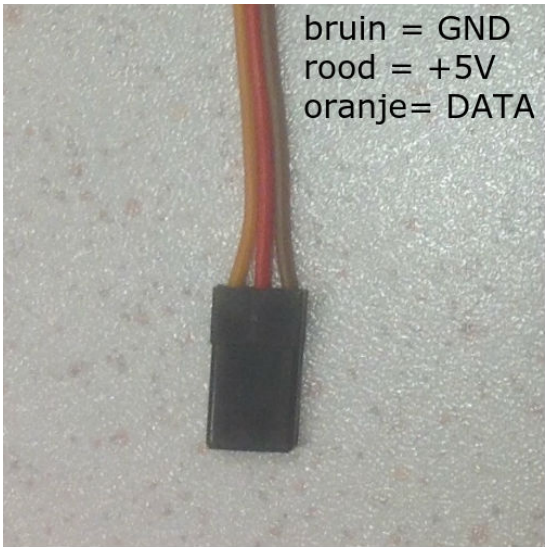
In deze les gaan we dingen laten bewegen. Bestuur een servo met een potmeter. Een servo is een bijzonder apparaatje. Het beweegt 180° in het rond en kan daarbinnen elke stand voor je aannemen. Voor de besturing heb je een bibliotheek nodig. Dit zit voor een servo standaard in de software, hierdoor wordt het schrijven van de sketch een stuk makkelijker. Je roept een servo op met: Sketch, Bibliotheek Importeren, Servo. Kijk ook eens bij de voorbeelden onder Bestand, Voorbeelden, Servo. Deze sketches zijn een paar voorbeelden hoe de servo gebruikt kan worden. Deze les is gemaakt uit het voorbeeld Knop. Voor apparatuur van Fun met Electronica gebruiken we het schema hieronder als hulpmiddel voor de draadjes.

Opdracht 8.1. Deze keer geen opdracht om de code te veranderen of het schema, maar een vraag: Wat voor implementaties kan je bedenken met een servo?

4.8.2 Afbeeldingen en Code



Afbeelding 34: Les 8 schema



Afbeelding 35: Draadjes

Code 29: 08_Servo_Pot.ino

```
1 /*
2   Weer een library die nodig is om gemakkelijk de servo te besturen.
3   Deze library staat automatisch in de Arduino IDE.
4  */
5 #include <Servo.h>
6
7 Servo myservo; // een variabele om de servo aan te sturen
8 int potpin = 0;// analoge pin wordt gebruikt door de potmeter
9 int val;       // lees de variabele waarde van de potmeter
10
11 void setup() {
12   myservo.attach(9);  // de servo is verbonden met pin 9
13 }
14
15 void loop()
16 {
17   val = analogRead(potpin);  // lees de waarde van de potmeter (waarde
       tussen 0 en 1023)
18   val = map(val, 0, 1023, 0, 179); // reken de waarde om voor de servo (
       waarde tussen 0 en 180)
19
20   myservo.write(val); // sets the servo position according to the scaled
       value
21
22   delay(15); // wacht tot de servo er is gekomen
23 }
```

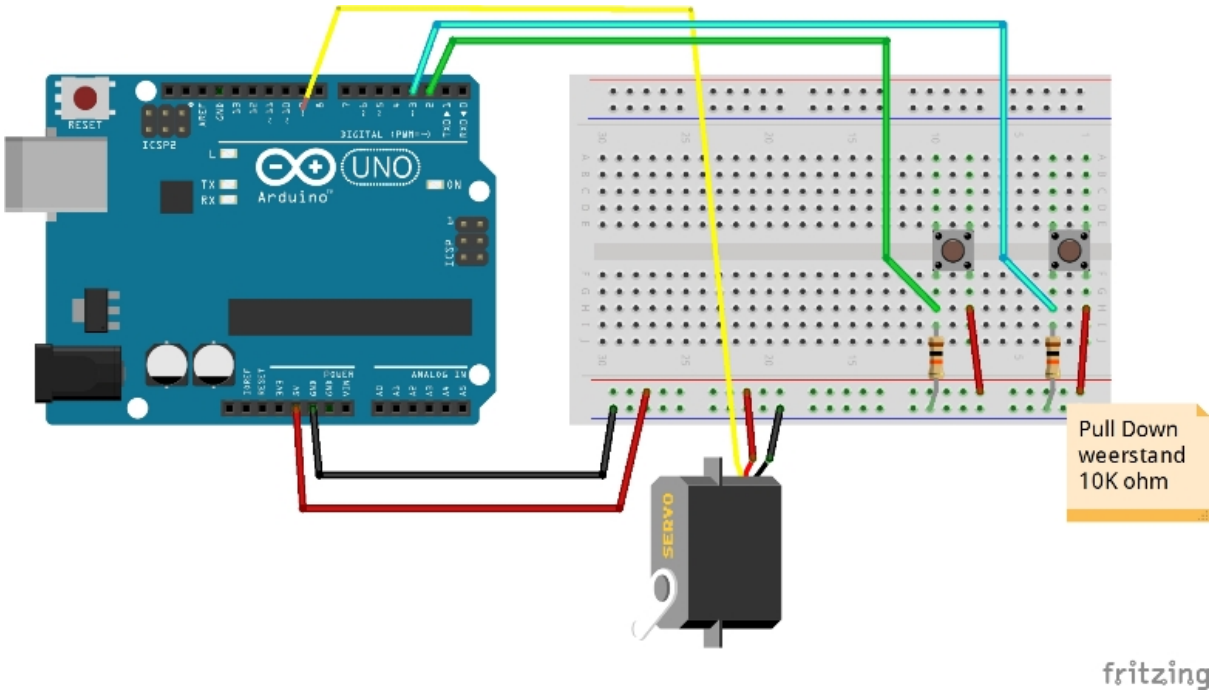
4.9 Les 9: Servo met knop

4.9.1 Functie en opdrachten

In deze les gaan we de servo bedienen met knoppen. De sketch lijkt op de sketch toneKeyboard. Hier gebruiken we de bibliotheek van de servo en de bewegingscommando's die bij servo's horen. Als een van de knoppen wordt ingedrukt gaat de servo naar 0° of 180°. De servo's van Fun met Electronica hebben dezelfde kleuren als de vorige les.

Opdracht 9.1. Maak een nieuwe toestand. Als er geen knop in wordt gedrukt moet de servo op 90° staan. De andere knoppen moeten hun eigen functie behouden.

4.9.2 Afbeeldingen en Code



Afbeelding 36: Les 9 schema

Code 30: 09_Servo_Knop.ino

```
1  #include <Servo.h>       // bibliotheek servo
2
3  Servo myservo;           //we noemen de servo myservo
4  int knop1 = 2;           //knop1 op pin 2
5  int knop2 = 3;           // knop2 op pin 3
6  int toestandknop1 = 0;   // de toestand van knop1 is een variabele
7  int toestandknop2 = 0;   // de toestand van knop2 is een variabele
8  int pos = 0;             // de positie van de servo is een variabele
9
10 void setup() {
11   pinMode (knop1, INPUT); // knop1 is een INPUT
12   pinMode (knop2, INPUT); // knop2 is een INPUT
13   myservo.attach(9);      // de servo is verbonden met pin 9
14 }
15
16 void loop()
17 {
18   toestandknop1 = digitalRead(knop1); // lees knop1
19   toestandknop2 = digitalRead(knop2); // lees knop2
20
21   if (toestandknop1 == HIGH) {        // als knop1 wordt ingedrukt
22     myservo.write(180);               // ga naar positie 180
23     delay(15);                        // wacht tot de servo er is
24   } else if (toestandknop2 == HIGH) { // als knop2 ingedrukt wordt doe iets
         anders..
25     myservo.write(0);                 // ga naar positie 0
26     delay(15);                        // wacht tot de servo er is
27   }
28 }
```

4.10 Les 10: DC motor

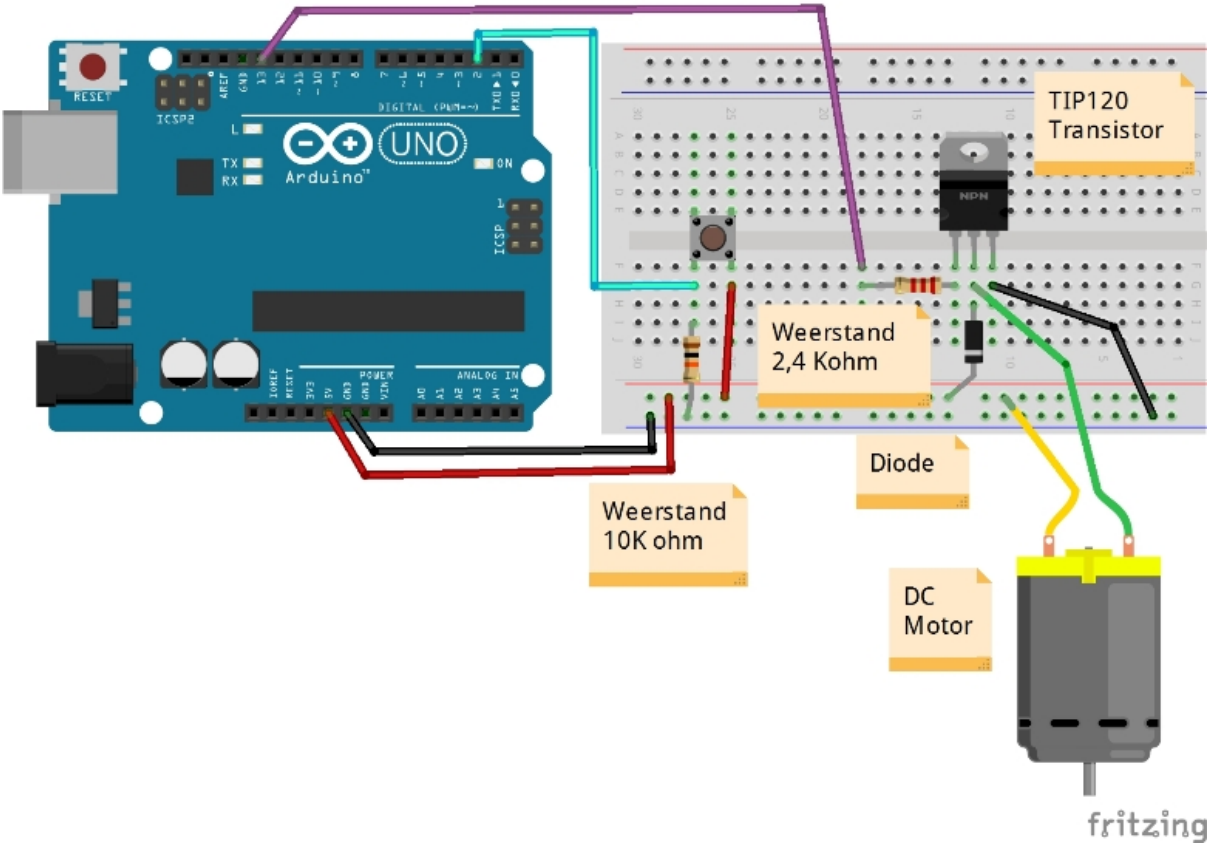
4.10.1 Functie en opdrachten

Deze sketch is het zelfde als die met de knop (les 3). Het verschil zit in wat en hoe je gaat schakelen. We schakelen een elektromotor met een transistorschakeling. Als we de motor direct aan de Arduino zouden aansluiten gaat de Arduino stuk, de transistor werkt hier als een soort hulpschakelaar.

Opdracht 10.1. Weer een vraag als opdracht: De motor draait nu 1n kant op. Hoe zorg je ervoor dat de motor de andere kant op draait? En bestaan er schakelingen die beiden richtingen kunnen besturen?

Opdracht 10.1. Hoe zou je de motor sneller of langzamer kunnen laten draaien?

4.10.2 Afbeeldingen en Code



Afbeelding 37: Les 10 schema

Code 31: 10_DC_Motor.ino

```
1  int knop = 2;          // knop aan pin 2
2  int motor =  13;       // motor aan pin 13
3  int toestandknop = 0; // variabele voor het lezen van de knop
4
5  void setup() {
6    pinMode(motor, OUTPUT); //motorpin is output
7    pinMode(knop, INPUT);   //knop is input
8  }
9
10 void loop() {
11   toestandknop = digitalRead(knop); // toestandknop is de waarde van knop
12   if (toestandknop == HIGH) {       // controleer of de knop ingedrukt is
13     digitalWrite(motor, HIGH);      // indien ingedrukt: motor aan
14   } else {
15     digitalWrite(motor, LOW);       // anders: motor uit
16   }
17 }
```

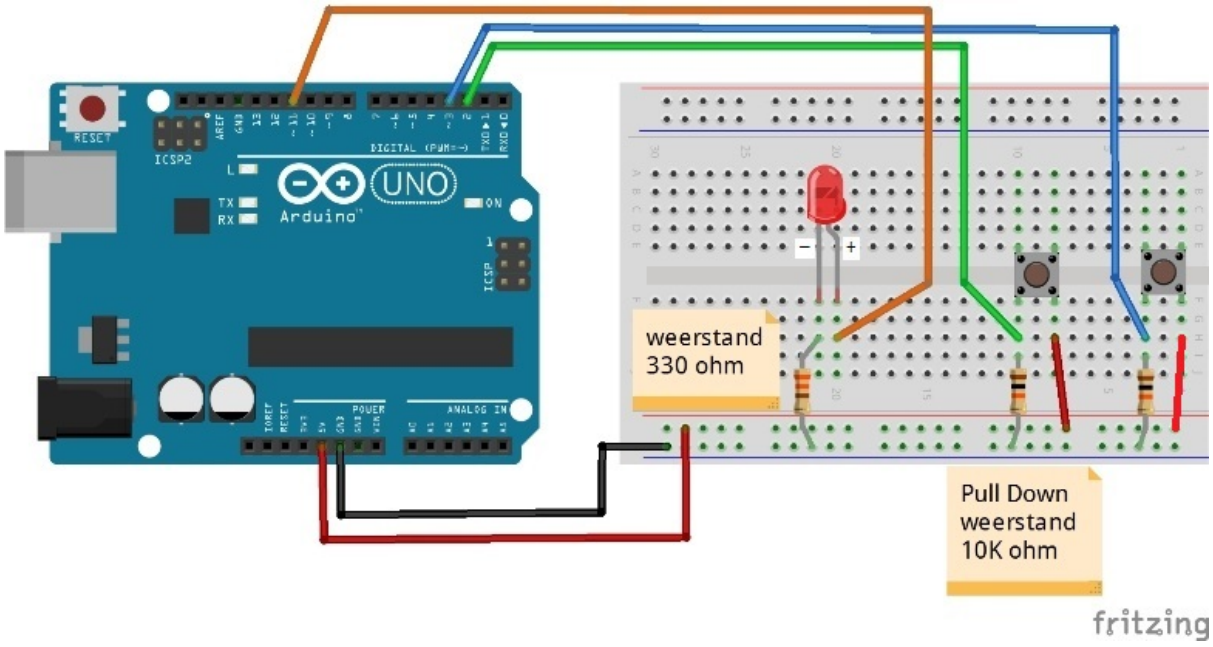
4.11 Les 11: digitalAnalog

4.11.1 Functie en opdrachten

Een knop bedienen op een Arduino noemen we een digitale ingang. Het is ingedrukt of niet. Een led laten branden is vaak een digitale uitgang, de led is aan of uit. Maar we kunnen met Arduino ook leds dimmen. Dit noemen we analogWrite. De uitgang is niet meer aan of uit, maar kan er ook tussen in zitten. Arduino doet dit met PWM. Alleen de poorten 3, 5, 6, 9, 10 en 11 kunnen analogWrite leveren. Deze poorten zijn te herkennen aan een slingertekentje. We gaan op een led twee knoppen aansluiten, 1 voor gedimde toestand en de ander voor vol vermogen.

Opdracht 11.1. In plaats van dat de knopjes een felheid geven, laat de lampjes (minder) feller laten worden om zo een soort alarm te simuleren.

4.11.2 Afbeeldingen en Code



Afbeelding 38: Les 11 schema

Code 32: 11_DigitalAnalog.ino

```
1  int knop1 = 2;           // knop aan pin 2
2  int knop2 = 3;           // knop aan pin 2
3  int led =  11;           // led aan pin 11
4  int toestandknop1 = 0;   // variabele voor het lezen van de knop
5  int toestandknop2 = 0;   // variabele voor het lezen van de knop
6
7  void setup() {
8    pinMode(led, OUTPUT);    // ledpin is output
9    pinMode(knop1, INPUT);   // knop is input
10   pinMode(knop2, INPUT);   // knop is input
11 }
12
13 void loop() {
14   toestandknop1 = digitalRead(knop1);    // toestandknop is de waarde van
       knop
15   toestandknop2 = digitalRead(knop2);    // toestandknop is de waarde van
       knop
16
17   if (toestandknop1 == HIGH) {           // controleer of de knop1 ingedrukt
       is
18     analogWrite(led, 50);                // indien ingedrukt: led aan op
         sterke 50 (waarde tussen 0 - 255)
19   } else if (toestandknop2 == HIGH) {    // controleer of de knop2 ingedrukt
       is
20     analogWrite(led, 255);               // indien ingedrukt: led aan op
         sterke 100 (waarde tussen 0 - 255)
21   } else {
22     digitalWrite(led, LOW);              // anders: led uit
23   }
24 }
```

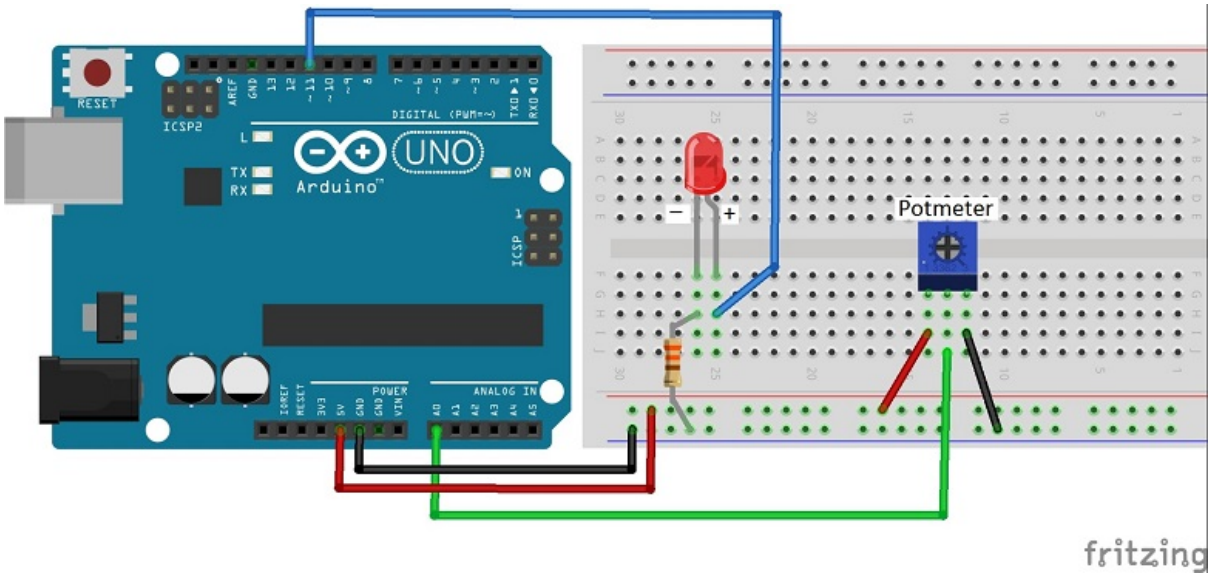
4.12 Les 12: Analog

4.12.1 Functie en opdrachten

Een vervolg op de vorige les zonder opdracht. Nu hebben we de ingang analoog gemaakt met een potmeter. We gaan het dimmen van de led de het analoge ingangswaarde koppelen. Deze les kan overgeslagen worden als de code duidelijk is zonder het systeem gebouwd te hebben.

Bij deze les zijn geen opdrachten geleverd. Probeer zelf iets te bedenken!

4.12.2 Afbeeldingen en Code



Afbeelding 39: Les 12 schema

Code 33: 12_Analog.ino

```
1  int led = 11; // led aan pin 11
2  int pot = A0; // potmeter aan A0
3  int val = 0;  // variable waarde van de potmeter
4
5  void setup() {
6    pinMode(led, OUTPUT);   // ledpin is output
7  }
8
9  void loop() {
10   val = analogRead(pot);      // lees de waarde van de potmeter
11   analogWrite(led, val / 4);  // waarde van de potmeter delen door 4
12 }
```

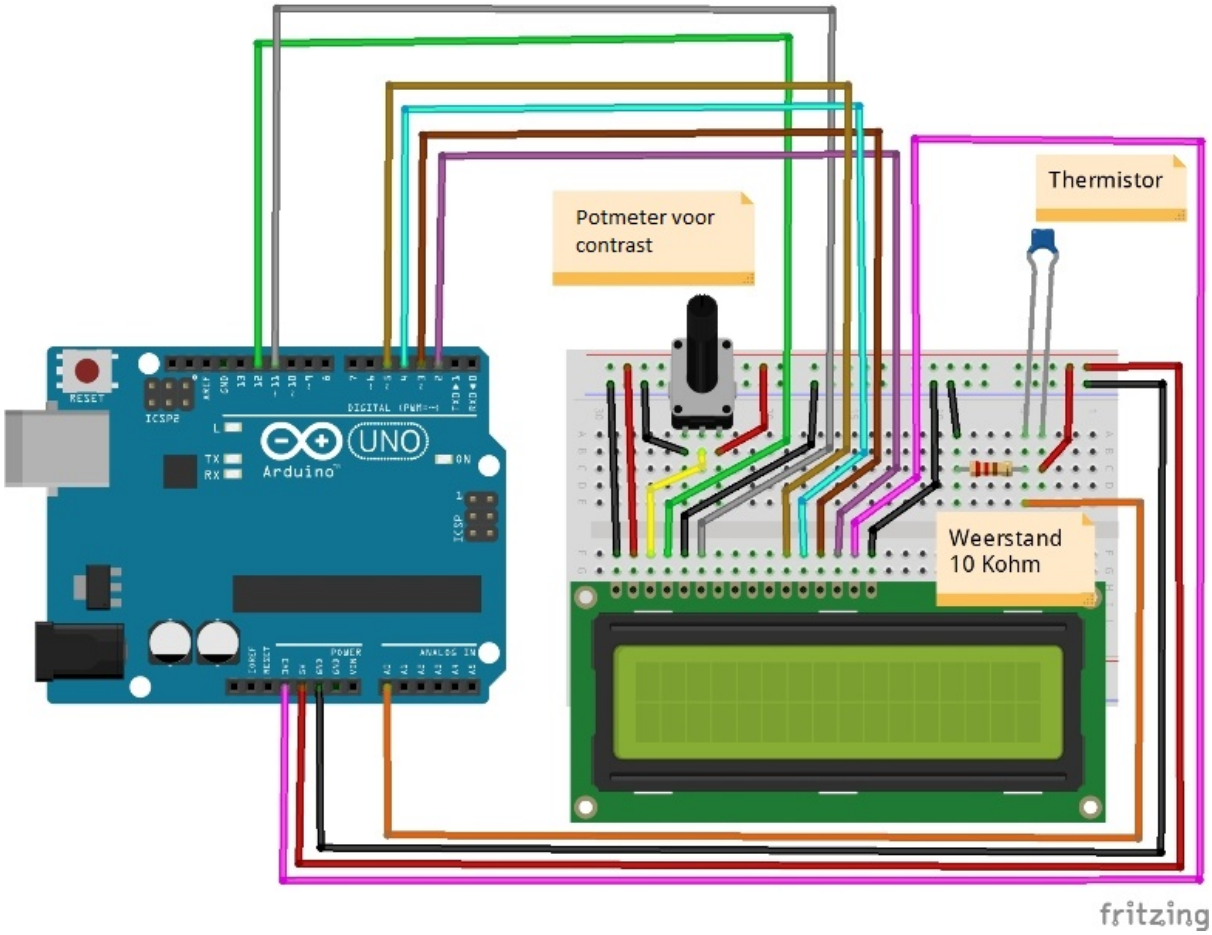
4.13 Les 13: Temperatuur op LCD

4.13.1 Functie en opdrachten

In deze les gaan we met een thermistor werken en een LCD schermpje. Een thermistor is een temperatuur gevoelige weerstand, we sluiten deze als spanningsdeler op. Voor LCD display moet je, net als bij de servo, een bibliotheek gebruiken. Deze zit standaard in de software opgenomen. Je roept hem op door te klikken op Sketch, Blibliotheek, Importeren, LiquidCrystal. Verder is in deze sketch gebruik gemaakt van een zelfgemaakt teken (CustomCharacter). Wil je meer weten over LCD display's kijk dan in de voorbeelden.

Opdracht 13.1. Zoek op het internet wat LiquidCrystal precies is. Zijn er nog meer functies die je kan gebruiken? Kan je een tekst van links naar rechts laten zweven?

4.13.2 Afbeeldingen en Code



Afbeelding 40: Les 13 schema

Code 34: 13_LCD.ino

```
1 #include <LiquidCrystal.h>  //bibliotheek van LCD
2
3 LiquidCrystal lcd(12, 11, 5, 4, 3, 2);   //hier is de LCD aangesloten
4
5 /*
6   functie voor het berekenen van de temperatuur. De waarde die we krijgen
       van de A0 pin zetten we om naar een temperatuur. Deze functies kunnen
       vaak terug gevonden worden op een datasheet.
7 */
8 double Thermister(int RawADC) {
9   double Temp;
10  Temp = log(((10240000 / RawADC) - 10000));
11  Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.0000000876741 * Temp
       * Temp * Temp));
12  Temp = Temp - 273.15; // Omrekenen Kelvin naar Celcius
13  return Temp;
14 }
15
16 /*
17  Variabelen voor een zelfgemaakte teken.   Het teken is een graden teken
18  */
19 byte graden[8] = {0b00111, 0b00101, 0b00111, 0b00000, 0b00000, 0b00000, 0
     b00000, 0b00000};
20
21 void setup() {
22  lcd.createChar(1, graden);  // maak het nieuwe character aan
23  lcd.begin(16, 2);           // zet het LCD scherm aan
24  lcd.clear();                // maak het LCD scherm leeg
25  Serial.begin(115200);       // begin de seriele poort
26 }
27
28 void loop() {
29  Serial.println(int(Thermister(analogRead(0)))); // lees en print
       thermometer op seriele monitor
30  lcd.print(int(Thermister(analogRead(0))));      // lees en print
       thermometer op LCD scherm
31
32  lcd.setCursor(4, 0);  // verander de positie waar geprint wordt op de LCD
33  lcd.write(1);         // print een waarde op LCD scherm
34
35  lcd.print("C");       // print C op het LCD scherm
36
37  delay(100);           // wacht 100 miliseconden
38  lcd.clear();          // leeg het LCD scherm
39 }
```

4.14 Les 14: Stappenmotor

4.14.1 Functie en opdrachten

Deze les moet je maken zonder hulptekening. Kijk goed in de sketch op welke pinnen je het driverboard moet aansluiten. De stappenmotor moet aangesloten worden op het bijgeleverde driverboard. Kom je niet aan uit? Vraag aan je docent de hulptekening. Die kan in de docentenhandleiding de tekening raadplegen. Werk je zonder de hulp van een docent? Vraag dan via `info@funmetelectronica.nl` de hulptekening op. Na het aansluiten zal de stappenmotor gaan lopen als je op een van de knoppen drukt.

Opdracht 14.1. Kom erachter wat er precies in de code gebeurd. Zoek op internet wat de stepper library heeft. Probeer een paar dingen te veranderen of toe te voegen.

4.14.2 Afbeeldingen en Code

Code 35: 14_Step.ino

```
1  #include <Stepper.h> // library
2  #define STAPPEN_PER_MOTOR_OMWENTELING 32
3  #define STAPPEN_PER_TOTAAL_OMWENTELING1 2048 // 1 omwenteling rechtsom
4  #define STAPPEN_PER_TOTAAL_OMWENTELING2 -1024 // 0.5 omwenteling linksom
5
6  Stepper small_stepper(STAPPEN_PER_MOTOR_OMWENTELING, 8, 10, 9, 11);  //
       stapenmotor variabele
7  int knop1 = 1;           // knop aan pin 1
8  int knop2 = 2;           // knop aan pin 2
9  int toestandknop1 = 0;   // knop1 variabele
10 int toestandknop2 = 0;   // knop2 variabele
11 int  TeMakenStappen;     // variabele voor aantal stappen
12
13 void setup(){
14   pinMode(knop1, INPUT);  //knop1 is input
15   pinMode(knop2, INPUT);  //knop2 is input
16 }
17
18 void loop(){
19   toestandknop1 = digitalRead(knop1);       // lees toestand knop1
20   toestandknop2 = digitalRead(knop2);       // lees toestand knop2
21
22   if (toestandknop1 == HIGH) {
23     TeMakenStappen  =  STAPPEN_PER_TOTAAL_OMWENTELING1;
24     small_stepper.setSpeed(500);            // zet de snelheid
25     small_stepper.step(TeMakenStappen);     // loop
26   }else if (toestandknop2 == HIGH) {
27     TeMakenStappen  =  STAPPEN_PER_TOTAAL_OMWENTELING2;
28     small_stepper.setSpeed(500);            // zet de snelheid
29     small_stepper.step(TeMakenStappen);     // loop
30   }
31 }
```

4.15 Les 15: Parkeersensor

4.15.1 Functie en opdrachten

Deze les moet je ook maken zonder hulptekening. Kijk goed in de sketch op welke pinnen je de onderdelen moet aansluiten. Kom je niet aan uit? Vraag aan je docent de hulptekening. Hij/zij kan in de docentenhandleiding de tekening raadplegen. Werk je zonder de hulp van een docent? Vraag dan via `info@funmetelectronica.nl` de hulptekening op. Bij het afmaken van de opstelling zal het lampje aan gaan als er een object dichtbij de ultrasoon afstandsmeter staat.

Opdracht 15.1. Maak de parkeersensor met een buzzer in plaats van een led.

Opdracht 15.2. Kan je net zoals in een auto het buzzer signaal sneller laten buzzen? En als het object te dichtbij is dat het een hoge constante piep geeft? Zo ja voer dit dan uit. Opgelet: enorm lastig!

4.15.2 Afbeeldingen en Code

Code 36: 15_Ultrasoon.ino

```
1  const int trigPin = 2;  // TRIG aan pin 2
2  const int echoPin = 4;  // ECHO aan pin 4
3  const int led = 13;     // LED aan pin 13
4  long duration;          // variabele voor de tijdslengte
5
6  void setup() {
7    pinMode (led, OUTPUT);    // LED is output
8    pinMode(trigPin, OUTPUT); // TRIG is output
9    pinMode(echoPin, INPUT);  // ECHO is input
10   Serial.begin(9600);       // start seriele monitor
11 }
12
13 void loop() {
14   digitalWrite(trigPin, LOW);         // LOW op TRIG geeft geen ultrasoon
       signaal
15   delayMicroseconds(2);               // wacht 2 mircoseconden
16   digitalWrite(trigPin, HIGH);        // begin het ultrasoon signaal
17   delayMicroseconds(10);              // wacht 10 microseconden
18   digitalWrite(trigPin, LOW);         // stop het ultrasoon signaal
19   duration = pulseIn(echoPin, HIGH);  // lees het signaal vertraging
20
21   Serial.print(duration); // geef de echo-tijd weer in de seriele monitor
22   Serial.println();       // print een regel verder
23   delay(100);             // wacht 100 miliseconden
24
25   if (duration < 1000) {  // als de echotijd kleiner is dan 100 dan...
26     digitalWrite (led, HIGH);   // zet de led aan
27   }else {                       // anders..
28     digitalWrite (led, LOW);    // zet de led uit..
29   }
30 }
```

4.16 Les 16: SOS met switch case

4.16.1 Uitleg Morse code

In de voorgaande lessen hebben we vooral gekeken naar wat je met een Arduino kan besturen en hebben we minder gelet op slimme oplossingen in het programma zelf. Deze les is speciaal bedoeld om je programma kleiner en overzichtelijker te maken. Dit kan je onder andere doen door te werken met switch-cases. Deze les is nieuw en daarom nog niet opgenomen in het geheel.

Morse Code. Morse is een communicatiecode, bestaande uit met tussenpozen uitgezonden signalen, die letters, leestekens en cijfers representeren. De code werd in 1835 uitgevonden en ontwikkeld door Samuel Morse met het doel deze te gebruiken voor de telegrafie. Morse kent twee symbolen: punten en streepjes, ofwel dits en dahs . De lengte van de 'dit' bepaalt de snelheid waarmee de boodschap wordt verzonden en wordt als 'eenheid' gebruikt. Een dah is volgens de afspraken drie keer zo lang als een dit. Spaties tussen dits en dahs binnen een lettercode hebben de lengte van één dit, spaties tussen letters in een woord hebben de lengte van 3 dits (één dah), en spaties tussen woorden zijn 7 dits lang.

In bovengenoemde tekst worden de woorden 'dit' en 'dah' genoemd, vaak gebruikt men hiervoor ook de woorden 'dot' en 'dash'. In de verdere uitleg gebruiken we 'dot' en 'dash'.

4.16.2 Implementatie 1

In deze les is de morse-code van SOS geprogrammeerd op pin 13. Deze pin is op de Arduino ingebouwd. De lengte van een dot is gekozen voor 500 miliseconden. Deze code is enorm lang en onnodig.

Code 37: SOS1.ino

```
1  int Led = 13;   // led op pin 13
2
3  void setup (){
4    pinMode (Led, OUTPUT);  // led is output
5  }
6
7  void loop () {
8    digitalWrite(Led, HIGH);  //dot
9    delay (500);              //dotlengte
10   digitalWrite(Led, LOW);   //symboolspatie
11   delay(500);               //symboolspatie lengte = dotlengte
12   digitalWrite(Led, HIGH);  //dot
13   delay (500);
14   digitalWrite(Led, LOW);
15   delay(500);
16   digitalWrite(Led, HIGH);  //dot
17   delay (500);
18   digitalWrite(Led, LOW);   //letterspatie
19   delay(1500);              //letterspatie lengte = 3 x dotlengte
20   digitalWrite(Led, HIGH);  //dash
21   delay (1500);             //dashlengte is 3x dotlengte
22   digitalWrite(Led, LOW);   //symboolspatie
23   delay(500);               //symboolspatie lengte = dotlengte
24   digitalWrite(Led, HIGH);  //dash
```

```
25   delay (1500);
26   digitalWrite(Led, LOW);
27   delay(500);
28   digitalWrite(Led, HIGH);  //dash
29   delay (1500);
30   digitalWrite(Led, LOW);   //letterspatie
31   delay(1500);
32   digitalWrite(Led, HIGH);  //dot
33   delay (500);
34   digitalWrite(Led, LOW);
35   delay(500);
36   digitalWrite(Led, HIGH);  //dot
37   delay (500);
38   digitalWrite(Led, LOW);
39   delay(500);
40   digitalWrite(Led, HIGH);  //dot
41   delay (500);
42   digitalWrite(Led, LOW);   //woordspatie
43   delay(3500);              //woordspatie lengt = 7x dotlengte
44 }
```

4.16.3 Implementatie 2

De code kan verbeterd worden door voor een dot en dash een eigen functie te maken. Ook wordt het interval gedeclareerd. Voor een letterspatie en een woordspatie zal tevens een functie gemaakt worden. Het wordt duidelijk dat de code al een stuk overzichtelijker is geworden.

Code 38: SOS2.ino

```
1  int Led = 13;       // led op pin 13
2  int interval = 500; // interval is 500 miliseconden
3
4  void setup () {
5    pinMode (Led, OUTPUT);  // led is output
6  }
7
8  void loop () {
9    dot();
10   dot();
11   dot();
12   letterSpatie();
13   dash();
14   dash();
15   dash();
16   letterSpatie();
17   dot();
18   dot();
19   dot();
20   woordSpatie();
21 }
```

```
22
23 /*
24   Functie voor dot.
25 */
26 void dot () {
27   digitalWrite(Led, HIGH);
28   delay (interval);
29   digitalWrite(Led, LOW);
30   delay(interval);
31 }
32
33 /*
34   Functie voor dash.
35 */
36 void dash () {
37   digitalWrite(Led, HIGH);
38   delay (interval * 3);
39   digitalWrite(Led, LOW);
40   delay(interval);
41 }
42
43 /*
44   Functie voor letterspatie.
45 */
46 void letterSpatie() {
47   delay (interval * 2);
48 }
49
50 /*
51   Functie voor woordspatie.
52 */
53 void woordSpatie() {
54   delay (interval * 6);
55 }
```

4.16.4 Implementatie 3

De code kan nog beter! In de 'void letter' maken we een switch naar 'karakters' (char) aan, met de variabele 'c'. Binnen de switch kennen we de cases toe. Die cases bestaan uit karakters en die plaatsen we tussen 'enkele – aanhalingsstreepjes'. Roep je in de hoofd – loop nu een bepaalde letter aan, dan gaat het programma binnen de switch – case zoeken. Heeft het programma de bepaalde letter gevonden, dan zorgt 'break' ervoor dat het programma doorgaat naar het einde van de 'void letter' loop. Daar staat een 'letterSpatie'. Vandaar dat in deze sketch de loop van 'woordSpatie' is aangepast.

Code 39: SOS3.ino

```
1 int Led = 13;        // led op pin 13
2 int interval = 500; // interval is 500 miliseconden
3
4 void setup () {
```

```
5   pinMode (Led, OUTPUT);
6 }
7
8 void loop () {
9   letter ('s');
10  letter ('o');
11  letter ('s');
12  woordSpatie();
13 }
14
15 /*
16   Functie voor dot.
17 */
18 void dot () {
19   digitalWrite(Led, HIGH);
20   delay (interval);
21   digitalWrite(Led, LOW);
22   delay(interval);
23 }
24
25 /*
26   Functie voor dash.
27 */
28 void dash () {
29   digitalWrite(Led, HIGH);
30   delay (interval * 3);
31   digitalWrite(Led, LOW);
32   delay(interval);
33 }
34
35 /*
36   Functie voor letterspatie.
37 */
38 void letterSpatie() {
39   delay (interval * 2);
40 }
41
42 /*
43   Functie voor woordspatie.
44 */
45 void woordSpatie() {
46   delay (interval * 4);
47 }
48
49 /*
50   Functie voor letter.
51   Kiest welke functie gekozen wordt voor een character.
52 */
53 void letter (char c) {
```

```
54   switch (c) {
55     case 's':                    // als de letter s is
56       dot(); dot(); dot();
57       break;
58     case 'o':                    // als de letter o is
59       dash(); dash(); dash();
60       break;
61   }
62   letterSpatie();
63 }
```

4.16.5 Opdracht

De switch case is enorm overzichtelijk. Er kan dus ook makkelijk letters aan toegevoegd worden. Probeer dit eens.