

Benodigdheden voor deze module

Offline	Online
Arduino	—
Breadboard	—
Jumperkabels	—
Weerstanden (220 Ω ; 330 Ω ; 1 k Ω ; 10 k Ω)	—
Potmeter (1 k Ω ; 10 k Ω)	—
Diode	—
2x LDR	—
LED (rood, geel, groen)	—
RGB LED	—
Servo motor	—
DC Motor	—
5x Drukknop	—
2N2222 transistor	—
LCD	—
Actieve piëzo element	—

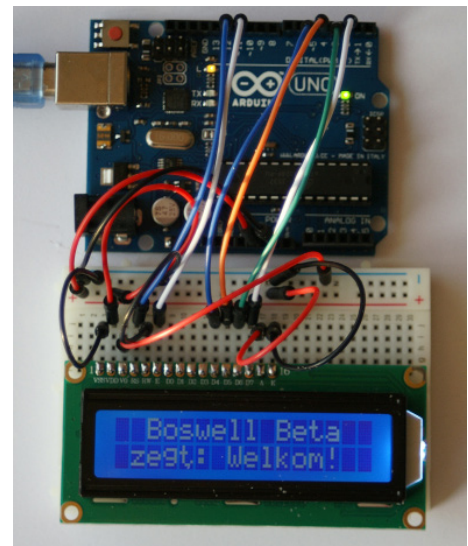
Benodigde voorkennis

Wet van Ohm	$R = \frac{U}{I}$
Serieschakeling	stroomsterkte gelijk
Parallelschakeling	spanning gelijk
Vermogen	$P = U \cdot I = I^2 \cdot R$
Werking dynamo/elektromotor	

Introductie

Leuk dat je aan de slag gaat met Arduino! Wellicht heb je nog geen idee wat een Arduino is en wat je er mee kunt. Een Arduino is een soort microcomputer. De Arduino sluit je aan op een computer waarbij je een programma (in Arduino noemen ze dat een sketch) stuurt naar de Arduino. De Arduino voert vervolgens je geschreven script uit en zorgt voor de uitvoer. Zo kun je een koelkast op een bepaalde temperatuur houden, een zelfrijdende robot aansturen, lichtsensoren maken, een lcd scherm op je shirt aansturen en ga zo maar door. Ook is het mogelijk om geen software te gebruiken en alleen met de elektronica te spelen. De Arduino is dan de spanningsbron.

In deze module gaan we aan het werk met de Arduino en leren we de basismogelijkheden van een Arduino. Bij het werken met een Arduino heb je twee belangrijke onderdelen: De Arduino en een breadboard. De Arduino is de computer, met invoer en uitvoer mogelijkheden. Op het breadboard sluit je de elektronica aan die aangestuurd wordt door de Arduino.

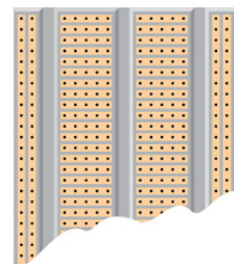
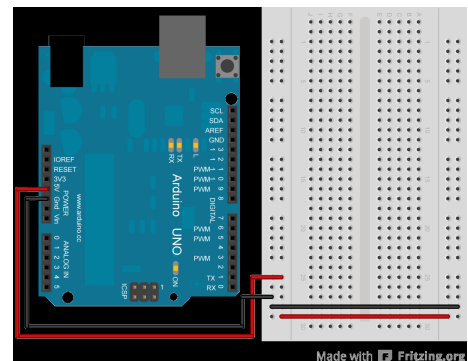


Breadboard

Het breadboard heeft aan beide zijdes twee kolommen die verbonden worden met de voeding (+ en -). De + kant sluit je aan op de 5 V uitgang of op een uitvoerpoort van de Arduino. De – kant sluit je aan op de GND (ground) van de Arduino. Alhoewel je de constante output niet altijd gebruikt, is het wel verstandig om deze altijd aan te sluiten.

Het breadboard heeft rijen en kolommen. Zie het figuur hiernaast voor een breadboard, en hieronder voor een opgewerkte breadboard. De punten in een rij zijn met elkaar verbonden. Maar laten we hier niet te lang bij stil staan...we gaan aan de slag!

NB: Vaak moet je even aangeven in welke USB-Poort je Arduino zit. Dit doe je door in het programma te gaan naar: Hulpmiddelen/Poort. Daar kun je de juiste USB poort aanklikken.



Opdracht 1 Het aansluiten van een LED

Een LED is een diode die licht uitzendt als er stroom door de LED gaat. De LED laat stroom maar door in één richting. De lange poot van de LED moet altijd op de + kant aangesloten worden, de korte poot op de – kant, zie het figuur hiernaast.

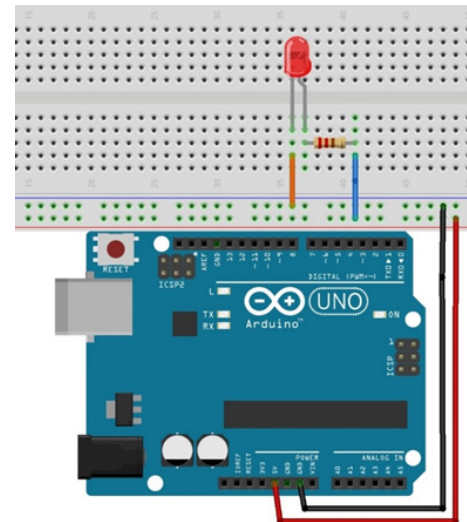
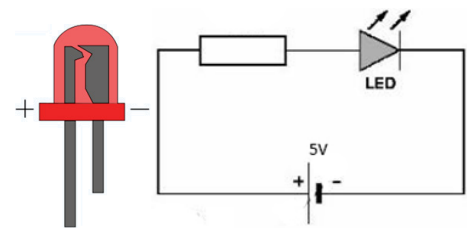
De stroom door een LED mag meestal maar 20 mA zijn. De spanning over de LED is dan ongeveer 2.0 V, deze waardes verschillen een beetje per soort LED, zie onderstaande tabel. De Arduino levert een 5.0 V spanning. De LED moet dus in serie geschakeld worden met een weerstand. De weerstand moet dus minimaal $150\ \Omega$ zijn ($R = \frac{U}{I} = \frac{3.0\text{ V}}{0.020\text{ A}} = 150\ \Omega$).

1. Er is geen weerstand aanwezig van $150\ \Omega$ maar wel een van $220\ \Omega$. Zoek de weerstand op met behulp van de kleurcode: de eerste ring moet rood zijn, de tweede ring ook en de derde ring bruin ($22 \cdot 10$). Een andere mogelijkheid is rood, rood, zwart, zwart ($220 \cdot 1$).
2. Sluit nu de 5 V uitgang van de Arduino aan op de + kolom en de GND (ground) uitgang van de Arduino aan op de – kolom.
3. Sluit de LED en de weerstand in serie aan, zie de tekening.
4. Verbind de Arduino via de usb met de computer. Als je het goed hebt gedaan brandt de LED!

Doordat de rode led bij lagere spanning licht uit zendt dan bijvoorbeeld een groene led, kun je een kleinere weerstand bij groene en blauwe led's gebruiken. Zo branden ze uiteindelijk toch nog even fel!

Kleur Drempelspanning

Blauw	2.3 V
Groen	2.0 V
Rood	1.5 V



KLEURCODE VAN WEERSTANDEN


KLEUR	1e RING	2e RING	3e RING	MULTIPL.	TOL.
ZWART	0	0	0	1	
BRUIN	1	1	1	10	± 1%
ROOD	2	2	2	100	± 2%
ORANJE	3	3	3	1k	
GEEL	4	4	4	10k	
GROEN	5	5	5	100k	± 0,5%
BLAUW	6	6	6	1M	± 0,25%
VIOLET	7	7	7	10M	± 0,10%
GRUJS	8	8	8		± 0,05%
WIT	9	9	9		
GOUD				0,1	± 5%
ZILVER				0,01	± 10%
BLANK					± 20%

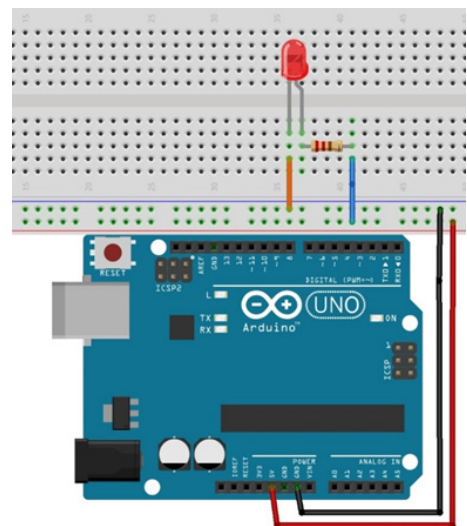
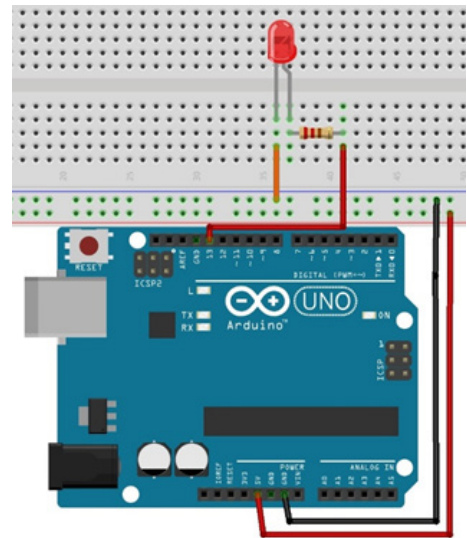
Opdracht 2 Een knipperende LED

Nu kun je je led wel laten branden, maar daar heb je nog niet veel aan... Een volgende stap is de LED aansturen met behulp van een stukje code. Om de LED aan te sturen met behulp van code moet je een uitvoerpoort (bijvoorbeeld pin 13) van de Arduino verbinden met je LED. De 5 V output heb je nu niet nodig, pin 13 levert nu de spanning. In de tekening zie je dat deze wel verbonden is, bij grotere projecten vergeet je snel de 5 V, vandaar!

- a) Bouw de opstelling die je ziet in de tekening en sluit de Arduino aan op de computer.

We willen de Arduino aansturen, daarvoor gebruiken we het Arduino programma.

- a) Open het programma en open het script blink via: bestand/voorbeelden/basis/blink.
- b) Controleer het script met het vinkje. Mocht het programma niet werken, dan geeft het onderaan een foutmelding weer (het kan zijn dat je de COM-poort moet toewijzen, dit doe je via hulpmiddelen/poort).
- c) Upload het script naar je Arduino met het teken  (snelcode: ctrl + u)
- d) Beschrijf kort wat je ziet. Probeer wat je ziet te verklaren met behulp van de code.
- e) Verander het script zodat de LED sneller knippert.
- f) Sluit drie verschillende LED lampjes aan, verander het script zodat ze om de beurt aan staan.
- g) Verander het script en maak een verkeerslicht waarbij oranje maar even aan staat.



Programmeren deel 1

Arduino heeft een eigen programma dat gebaseerd is op C++. Heb je al ervaring met programmeren en/of met C++ (of java), dan is het programmeren heel makkelijk. Heb je geen ervaring met programmeren? Het is veel minder moeilijk dan je misschien wel denkt!

Bovenaan het script Blink staat `/*` dit betekent dat alles na dit beginteken en voor het afsluitteken `*/` commentaar is. Hier zet je neer hoe het programma heet, wat het doet, wie het gemaakt heeft en wanneer je het voor het laatst hebt veranderd.

```
Blink$
1  /*
2   * Blink
3   * Turns on an LED on for one second, then off for one second, repeatedly.
4   * Most Arduinos have an on-board LED you can control. On the Uno and
5   * Leonardo, it is attached to digital pin 13. If you're unsure what
6   * pin the on-board LED is connected to on your Arduino model, check
7   * the documentation at http://www.arduino.cc
8   *
9   * This example code is in the public domain.
10  */
11
12 // the setup function runs once when you press reset or power the board
13 void setup() {
14   // initialize digital pin 13 as an output.
15   pinMode(13, OUTPUT);
16 }
17
18 // the loop function runs over and over again forever
19 void loop() {
20   digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
21   delay(1000);           // wait for a second
22   digitalWrite(13, LOW);  // turn the LED off by making the voltage LOW
23   delay(1000);           // wait for a second
24 }
25 }
```

Een tweede mogelijkheid om commentaar toe te voegen is met behulp van `//`. Alles op dezelfde regel na `//` is commentaar. Zo kun je bijhouden wat de code op die regel doet. Met behulp van de snel code `ctrl /` kun je code snel omzetten naar commentaar.

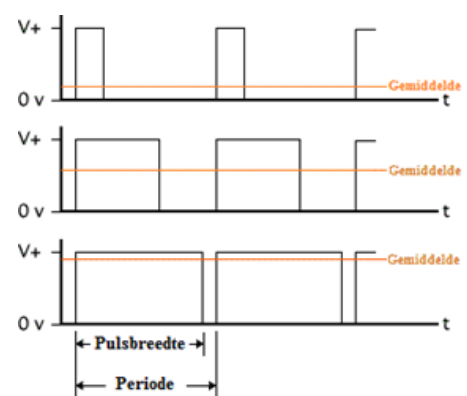
Voordat de belangrijkste code wordt uitgevoerd moet je zeggen wat er precies aangestuurd wordt. We geven aan dat in pin 13 iets zit en dat de Arduino daar een output (spanning) geeft als jij dat wilt. Iets netter zou zijn om nog voor de setup aan te geven hoe pin 13 heet: je geeft de pin een naam (`int LEDrood = 13;`). Wil je pin 13 vervolgens aansturen, dan kan je dat doen met de naam LEDrood.

Alles wat tussen de accolades `{}` van de loop staat wordt continu herhaald. Eerst wordt pin 13 hoog (5.0 V) gemaakt met behulp van de code `digitalWrite` (`digitalWrite` kan alleen aan of uit). Daarna moet het programma 1000 ms wachten (delay) voordat de volgende regel code wordt uitgevoerd. De volgende regel code maakt pin 13 weer laag (0.0 V). Daarna moet het programma opnieuw 1000 ms wachten, waarna het de loop opnieuw start.

De pin wordt aangestuurd met een hoog of met een laag signaal. Zit daar nog iets tussen? Ja en nee...De output is altijd 0.0 V of 5.0 V. Maar je kunt de LED wel dimmen door maar een bepaalde tijd de LED aan te zetten. Als de LED snel genoeg knippert zie je niet dat de LED knippert, het lijkt er alleen op dat de LED minder fel brandt. Wanneer je een LED wilt dimmen gebruik je een output met het symbool `~`. Dit is een Puls Width Modulation (PWM). De waarde van de PWM zit tussen de 0 (geheel uit) en 255 (geheel aan).

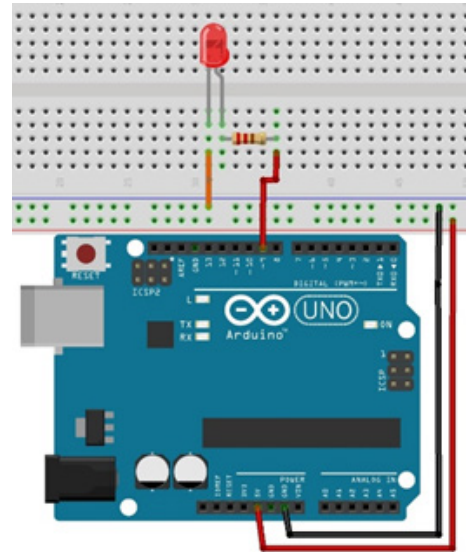
```
void setup(){
  pinMode(13, OUTPUT);
}
```

```
void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```



Opdracht 3 Een LED dimmen

- Bouw de opstelling die hiernaast staat. Gebruik bij de output een PWM pin, bijvoorbeeld Pin 9 (let op, de LED hoeft niet aangesloten te worden aan de constante spanning, de spanning wordt nu geleverd door pin 9).
- Open het script Fade in de voorbeelden/basis en upload het script.
- Beschrijf wat je ziet en probeer met behulp van de code een te verklaren wat er gebeurt.
- Verander de code zodat de LED sneller volledig brandt en sneller uit is. Let op, er zijn twee manieren! Probeer ze allebei uit.
- In de code staat `analogWrite`. Voorheen hebben we `digitalWrite` gebruikt. Leg uit waarom dat nu niet kan.

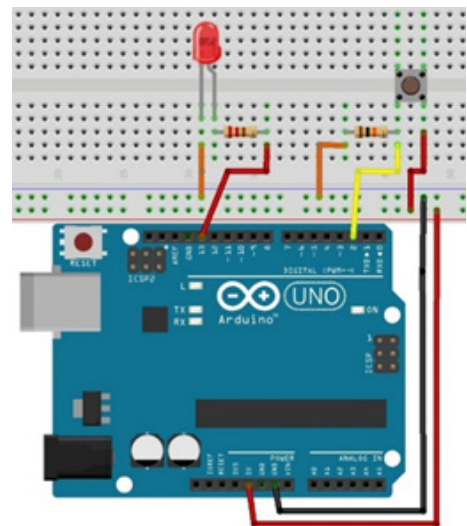


Opdracht 4 Een aan en uit knop voor de LED

We kunnen nu de LED met behulp van de code aan en uit zetten en zelfs dimmen. Maar vaak wil je ook een schakeling handmatig aan en uit kunnen zetten. Daarvoor hebben we een drukknop nodig.

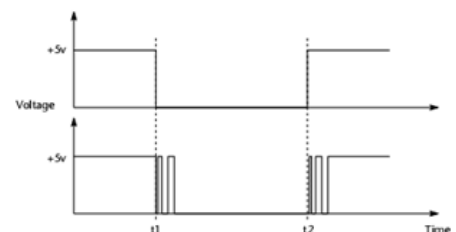
- Bouw de schakeling die hiernaast staat. Let op dat je een grote weerstand gebruikt zodat de te leveren stroom niet te groot is, een Arduino is namelijk slecht in het leveren van grote stroomsterktes.

Het idee is nu dat we pin 2 gebruiken als INPUT. Pin 2 meet de spanning op dat punt (vergelijkt deze met 0.0 V). Dit gebeurt met de code `digitalRead()`. Deze kan nu een hoog (5.0 V) of laag (0.0 V) signaal meten. Met een analoge pin zoals A0 t/m A5 kunnen ook tussen gelegen volages gemeten worden, met 10 bits resolutie. Pin 2 meet alleen een spanning als de knop (button) ingedrukt wordt.



- Open het script Button via voorbeelden/digitaal en upload het script.
- Druk op de knop en controleer wanneer de LED uit is en wanneer deze aan is.
- Pas het script aan zodat de functie van de knop precies omgedraaid wordt.

NB: Met de button is er iets bijzonders aan de hand. Deze heeft namelijk een zogenaamde Bounce. Dit betekent dat de spanning niet direct van 0 V naar 5 V gaat maar, nog een keer op en neer gaat. Dit komt omdat er in de button een veer zit die op en neer gaat. Bij het gebruik en uitlezen van de button is het dus handig om een delay in te bouwen...



Programmeren deel 2

Het script Button is uitgebreider dan we voorheen hebben gezien. We lopen stap voor stap door het script.

We hebben te maken met twee poorten waar iets moet gebeuren. Pin 2 is een invoer en pin 13 moet een uitvoer zijn. De pin verandert niet, dit is een constante (`const`). Het is een gehele waarde, een integer (`int`). We geven pin 2 een herkenbare naam, pin 2 heet nu `buttonPin`. Pin 13 heeft de naam `ledPin` gekregen.

We willen straks weten wat de 'staat' van de knop is (ingedrukt of niet). Deze kan 1 of 0 zijn. We moeten even vertellen dat we de staat van de knop willen weten (aanmaken van een variabele) en deze straks willen vergelijken. Voordat we het script laten draaien is de `buttonState` gelijk aan 0.

Zoals gezegd, we moeten even vertellen dat de `ledPin` een `OUTPUT` is en de `buttonPin` een `INPUT`. Zo, dat weten ze nu dan ook...

Er moet iets gebeuren als de knop wordt ingedrukt. Aan het begin van de loop wordt de knop uitgelezen (eigenlijk wordt er alleen gecontroleerd of er een 5.0V spanning over de weerstand staat). Daarna volgt een if-statement. Als (`if`) de knop is ingedrukt (`buttonState == HIGH`) dan moet de LED gaan branden (`digitalWrite(ledPin, HIGH)`). In alle andere gevallen (`else`), moet de LED niet branden (`digitalWrite(ledPin, LOW)`).

Het is een makkelijk if-statement. Je kunt ook meerdere voorwaarden stellen. Als je twee knoppen tegelijk in moet drukken voordat de LED gaat branden zet je `&&` tussen de tekens (`if buttonState1 == HIGH && buttonState 2 == HIGH`).

Je kunt ook het teken `||` gebruiken. Dan moet of de ene voorwaarde gelden of de andere. Helaas blijft het lampje nu nog niet branden als je de knop hebt ingedrukt. Je kunt er wel zelf voor zorgen dat je met één knop het lampje aan en uit kan zetten.

Vergeet niet de nieuwe variabele (`int`) te definiëren aan het begin! Er zijn nog eenvoudigere manieren, deze zijn alleen getoond als script. Leg uit hoe ze werken!

```
const int buttonPin = 2;
const int ledPin = 13;
```

```
int buttonState = 0;
```

```
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
```

```
void loop() {
  buttonState =
    digitalRead(buttonPin);
  if (buttonState == HIGH) {
    digitalWrite(ledPin,
      HIGH);
  }
  else {
    digitalWrite(ledPin,
      LOW);
  }
}
```

```
void loop() {
  buttonState =
    digitalRead(buttonPin);
  if (buttonState == HIGH
    && state == LOW) {
    state2 = HIGH;
  }
  if (buttonState == HIGH
    && state == HIGH) {
    state2 = LOW;
  }
  state = state2;
  digitalWrite(ledPin,
    state);
  delay (100);
}
```

```
if(buttonState == HIGH){i
  ++;}
digitalWrite(ledPin,i%2);
```

```
digitalWrite(13,!
  digitalRead(13));
```

Opdracht 5 Het beveiligen van een kluis

De kluis van de bank is beveiligd. De kluis gaat pas open (de LED brandt pas) als er twee sleutels worden omgedraaid (knoppen ingedrukt). Een sleutelgat bevindt zich op de kamer van de directeur, de andere zit naast de kluis.

Bouw de bijbehorende schakeling en programmeer het script zodanig dat de kluis pas open gaat als beide sleutels tegelijk worden omgedraaid.

Opdracht 6 Een wisselschakeling

Als je onderaan de trap staat wil je het licht bovenaan de trap aan zetten. Bovenaan de trap zit nog een lichtknop. Daarmee kan je het licht weer uit of aan zetten. Bouw de schakeling met twee knoppen en schrijf het programma zodat je een werkende wisselschakeling hebt. Lukt dit niet meteen? Probeer het dan eerst met een enkele knop.

Opdracht 7 Een voetgangersoversteekplaats

Voor voetgangers is een speciale overgangsplek gemaakt. Deze plek ligt aan een drukke weg. Als er geen voetgangers zijn, staat het verkeerslicht voor de auto's op groen. Op het moment dat een voetganger over wil steken, drukt deze op de knop. Het verkeerslicht voor de auto's gaat dan van groen naar geel naar rood. Voetgangers hebben dan 15 seconde om over te steken. Het verkeerslicht voor hen staat op groen! Dan gaat het groene lampje vijf keer knipperen en springt op rood. Een seconde later springt het verkeerslicht voor de auto's weer op groen.

Maak dit systeem.

Elektronica deel 1

Wanneer je alleen een knop en een LED gebruikt ben je snel uitgekeken en heb je nog nauwelijks iets gedaan... We willen robots aansturen, racemachines niet tegen muren laten botsen, kassen automatisch aansturen en ga zo maar door. Daar hebben we sensoren voor nodig. Vrijwel alle sensoren werken hetzelfde, de weerstandswaarde verandert bij een veranderende input (bijvoorbeeld licht of kracht). We gaan beginnen met een lichtsensor. Voor een lichtsensor hebben we een LDR nodig, zie de foto. LDR staat voor Light Dependent Resistor (licht afhankelijke weerstand). De weerstandswaarde verandert met veranderende lichtintensiteit, zie de grafiek.

We sluiten de LDR in serie aan met een Ohmse weerstand (een weerstand met een constante weerstandswaarde). Zo hebben we een spanningsdeler gemaakt. Een deel van de spanning staat over de LDR en een deel van de spanning staat over de weerstand. Het enige wat we moeten doen is de spanning over de LDR uitlezen en we weten hoe licht het is!

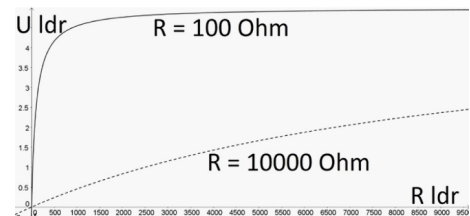
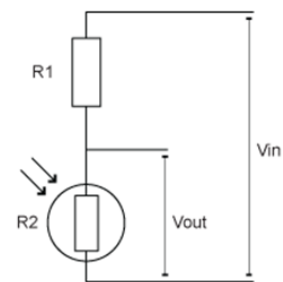
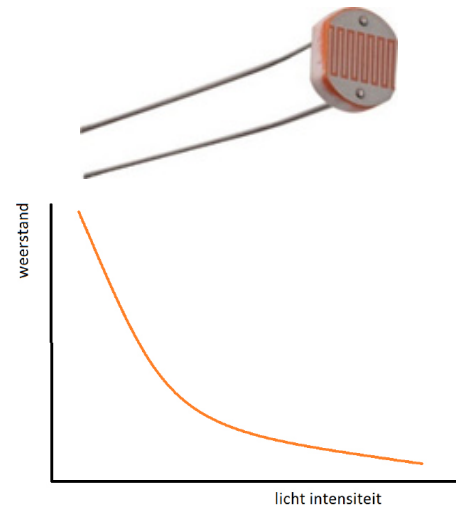
Het is belangrijk om iets meer te weten over hoe een spanningsdeler werkt: bij het uitlezen van een sensor maken we gebruik van een spanningsdeler. De LDR en de Ohmse weerstand staan in serie zodat er geldt: $R_{\text{totaal}} = R_{\text{LDR}} + R_{\Omega}$. De bronspanning is 5.0 V en omdat de weerstanden in serie geschakeld zijn geldt dat de stroomsterkte overal gelijk is.

De stroomsterkte bereken je met: $I = \frac{U_{\text{bron}}}{R_{\text{totaal}}}$.

De spanning wordt netjes verdeeld en de grootste weerstand krijgt de meeste spanning. Als je deze informatie combineert zie je dat de verhouding tussen de spanningen gelijk is aan de verhouding tussen de weerstanden: $\frac{U_{\text{LDR}}}{U_{\Omega}} = \frac{R_{\text{LDR}}}{R_{\Omega}}$.

Als het lichter is, wordt de weerstandswaarde van de LDR kleiner waardoor er minder spanning over de LDR staat maar meer spanning over de Ohmse weerstand. De spanning over de LDR ($U_{\text{LDR}} = 5.0 \text{ V} \cdot \frac{R_{\text{LDR}}}{(R_{\text{LDR}} + R_{\Omega})}$) kun je meten met behulp van de ANALOG IN van de Arduino. De spanning is dan een maat voor de gemeten lichtintensiteit.

In welk gebied je sensor gevoelig moet zijn bepaalt de keuze voor een Ohmse weerstand. Zie de tweede grafiek. De LDR heeft een waarde tussen de 10 kΩ en 20 kΩ: dus kies een grote weerstand!



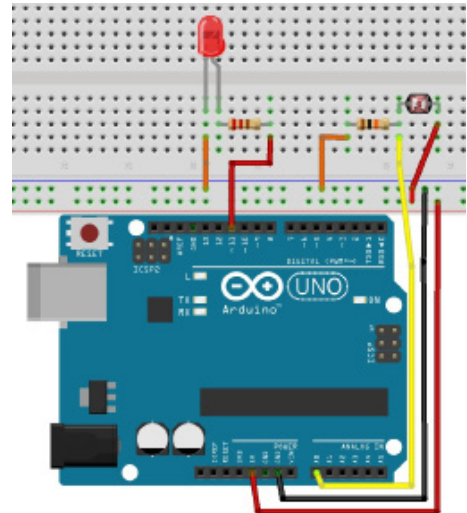
Opdracht 8 Een lichtsensor

De schakeling lijkt erg op de schakeling van opdracht 4. Alleen nu gebruiken we een LDR en een ANALOG IN.

- Bouw de schakeling.
- Open het script `analogReadSerial`: `voorbeelden/basis/Analogreadserial`.

De belangrijkste code vind je hieronder.

```
void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
  delay(10);
}
```



De Arduino krijgt de opdracht om de analoge poort uit te lezen (`analogRead(A0)`). Deze waarde wordt gehangen aan de variabele `sensorValue`.

Vervolgens willen we deze waarde weten. De waarde wordt dan ook geprint voor ons (`Serial.println(sensorValue);`), deze is te lezen met behulp van de seriële monitor (het loepje rechts bovenin). Voor de stabiliteit is het goed om een `delay` in te bouwen.

- Upload het script naar de Arduino en lees de waarden uit met behulp van de Serial Monitor.
- Bedek met je hand de LDR. Verandert de gegeven waarde?
- Combineer opdracht 3 en deze opdracht. Zorg ervoor dat de LED feller gaat branden als het donkerder wordt.
- Breid de schakeling verder uit zodat je de hele schakeling ook met een knop aan en uit kan zetten.
- Leg met behulp van grafiek 2 uit dat een grote waarde van R_{Ω} ervoor zorgt dat de LDR in het hele bereik vrijwel even gevoelig is.
- Bedenk een manier om de sensor zelf te laten ijken. Dus laat de sensor zelf de minimale en maximale waarde bepalen.

Opdracht 9 Reactietijd meten

Bouw een reactietijdmetr waarbij iemand zo snel mogelijk op de knop moet drukken als een rode LED uit gaat en een groene LED aangaat. Om dit lastiger te maken voor de persoon die moet drukken kun je gebruik maken van functie `random(a, b)`. Waarbij a en b getallen zijn, zo kan je een willekeurige delay inbouwen. De functie `millis()` geeft aan hoeveel milliseconde ervoor bij zijn gegaan.

Programmeren deel 3

De Arduino kan niet alle waarden uitlezen. De ANALOG IN heeft een 10 bits chip. Dit betekent dat er $2^{10} = 1024$ waarden doorgegeven kunnen worden. Het is alsof je de 5.0 V in 1024 kleine blokjes verdeelt. De waarde die je uitleest bij opdracht 8 is dus ook niet de spanning zelf, maar het getal dat bij een spanning hoort. Het getal 223 hoort dus bij:

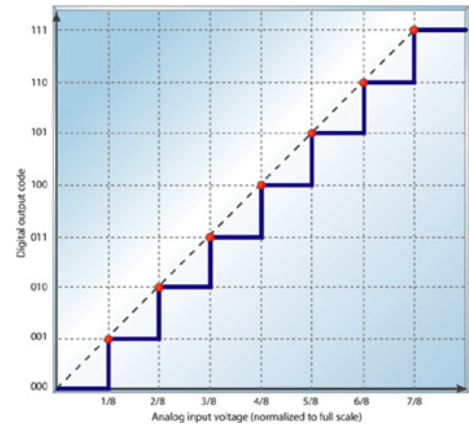
$\frac{223}{1023} \cdot 5.0 \text{ V} = 1.09 \text{ V}$. De PWM (~) is een 8 bits systeem en kan dus $2^8 = 256$ waarden geven.

Oei...zie je het probleem? We kunnen lezen met 1023 verschillende waardes maar schrijven kan maar met 255 waardes.... Gelukkig is er een code die zorgt voor automatische schalen (**map**).

De functie **map** wil 5 getallen hebben. Het eerste getal is de waarde die de analoge poort uitleest. Het tweede getal is het laagste getal dat uitgelezen kan worden, het derde getal de grootste waarde die uitgelezen kan worden. Dit hoeven niet per se 0 en 1023 te zijn, de waarde verkrijgt je bij het iken van je sensor. De gevoeligheid van je sensor kan dus ook zitten tussen 500 en 900, zie wederom grafiek 2 uit programmeren deel 3. Het vierde en vijfde getal geven het bereik van de output weer. In het voorbeeld moet een input waarde van 500 een output waarde worden van 255. Een input waarde van 900 moet een output waarde van 0 worden.

Een functie die ook handig is, is de functie ++. Elke keer als de loop de volgende cyclus ingaat, wordt de waarde met 1 verhoogd. Zo kun je bijvoorbeeld eenvoudig bijhouden hoeveel loops (iteraties) er al zijn geweest. Ook is het mogelijk om op die manier elke keer een andere pin aan te sturen of later een frequentie te veranderen.

Als ++ bestaat, dan zal -- ook wel bestaan. En dat klopt. -- zorgt ervoor dat de waarde met 1 wordt verlaagd. Beide codes kunnen in een **for**-loop geplaatst worden. Dit is een loop binnen de loop. Kijk eens naar het voorbeeld waarbij er een soort aftel mechanisme in zit voordat het programma zelf begint...(let op, de variabele **i** moet je nog wel even declareren voor de setup!



```
void loop() {  
    sensorValue = analogRead  
        (sensorPin);  
    brightness = map(  
        sensorValue  
        ,500,900,255,0);  
    Serial.println(  
        sensorValue);  
    delay(10);  
    analogWrite(ledPin,  
        brightness);  
}
```

```
void loop(){  
    for(i<5;i++;) {  
        digitalWrite(LEDPin,  
            HIGH);  
        delay(100-i);  
        digitalWrite(LEDPin,  
            LOW);  
        delay(100);  
    }  
    [...]  
}
```

Opdracht 10 Sneller programmeren

Herschrijf opdracht 2h met de functie ++.

Opdracht 11 Alarmschakelaar

Onder de kassa van een winkel zit een schakelaar (drukknop). Als deze wordt ingedrukt dan moet er een pulserend alarmbel (ditmaal nog een LED) gaan. Maak dit systeem.

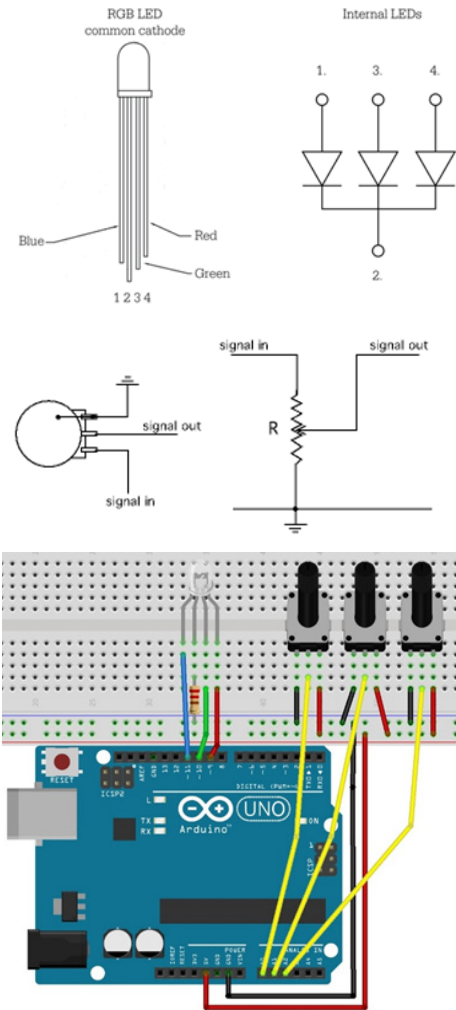
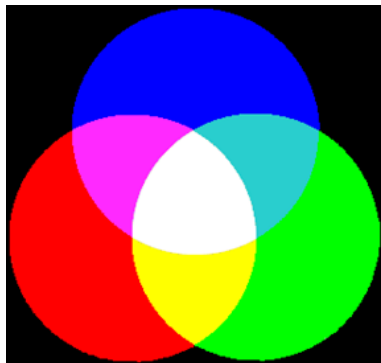
Opdracht 12 Een RGB LED

Een RGB LED heeft vier pootjes. In de LED zitten eigenlijk drie LED's verwerkt. Door de drie LED's met behulp van een PWM (~) aan te sturen kun je kleuren mengen en zo tussenliggende kleuren krijgen. Nu kunnen we via de software de sterkte van elke kleur bepalen, maar het zou leuk zijn wanneer we ook handmatig de kleuren kunnen instellen. We gebruiken hiervoor variabele weerstanden ook wel potmeters genaamd.

De potmeter heeft drie poten. Eentje voor de 5.0 V in, eentje voor de aarde (0.0 V) en een poot voor het uitlezen van de meter met behulp van een analoge pin, zie ook de figuur hiernaast.

Als je de draaiknop van de potmeter draait, verandert de weerstand, dit verandert dus ook de waarde van de spanning die je uit leest. Een potmeter is dus eigenlijk een spanningsdeler.

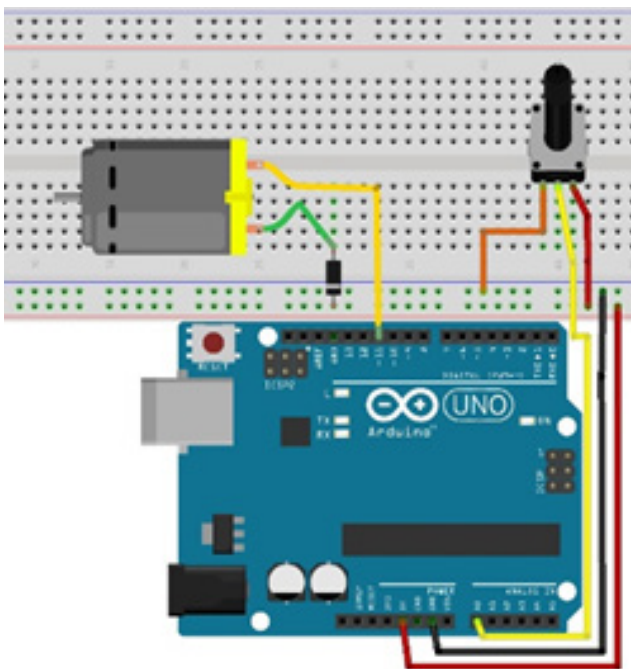
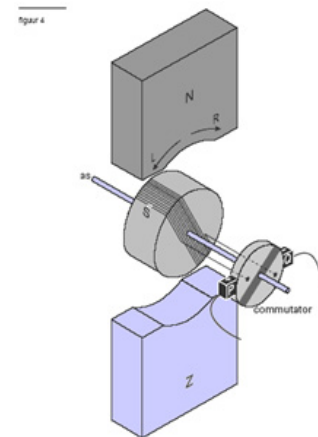
- Bouw de schakeling.
- Schrijf de bijbehorende code waarbij je eerst de waarde van de potmeter uitleest en vervolgens de RGB LED aanstuurt.
- Probeer de verschillende standen van de RGB Led en controleer of het plaatje hieronder goed overeenkomt met de kleuren van de LED.



Opdracht 13 De snelheid van een elektromotor

Een elektromotor zorgt ervoor dat van elektrische energie bewegingsenergie gemaakt wordt. Een elektromotor vind je in alle apparaten die bewegen en werken op elektriciteit. De snelheid van een elektromotor is afhankelijk van de spanning over de motor. We werken in deze les dus met de PWM pin (\sim). We kunnen natuurlijk direct de snelheid van de elektromotor in ons script zetten, maar het zou handiger zijn wanneer we de snelheid van de elektromotor kunnen aanpassen, bijvoorbeeld met behulp van een draaiknop. We gaan gebruik maken van een variabele weerstand, ook wel een potmeter genoemd.

- Bouw de rechterzijde van de schakeling met de potmeter. De elektromotor hoeft je nog niet aan te sluiten.
- Leg uit waarom de signal out van de potmeter naar een ANALOG IN moet.
- Leg uit waarom de diode is opgenomen in de schakeling.
- Programmeer het bijbehorende script zodat de snelheid van draaien van de elektromotor ingesteld kan worden met behulp van de potmeter. Gebruik daarbij de map functie.

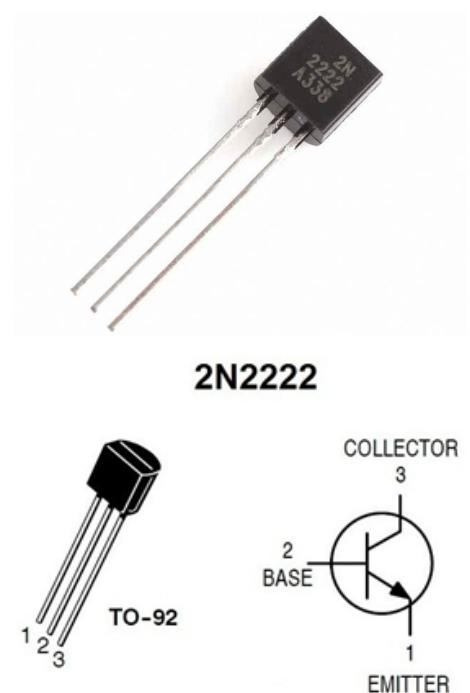


Elektronica deel 2

Een elektromotor kan veel stroom vragen. De Arduino is slecht in het leveren van veel stroom. Om toch grote stroomvragers te voorzien van de benodigde stroom kun je gebruik maken van een externe spanningsbron zoals een batterij. Dan heb je nog wel een transistor nodig zodat je de grootte van de stroom kunt regelen en je de snelheid van draaien van de elektromotor kunt aanpassen.

De transistor (hier gebruiken we alleen een zogenaamde NPN-transistor) heeft drie pootjes, zie de foto. Links zit de emitter, in het midden de base en aan de rechterzijde de collector. (LET OP! Bij verschillende transistors wordt de volgorde omgedraaid, wij hebben de 2N222 transistor). Ten alle tijden wordt aan de collector de externe voeding (+ zijde) verbonden. Spanning over de base bepaalt of er stroom gaat lopen naar de emitter. Daarmee is een transistor een soort van kraan: de hoeveelheid spanning bij de base bepaalt de hoeveelheid doorgelaten stroom. De base moet daarom aangestuurd worden met een PWM pin als je de snelheid wilt regelen.

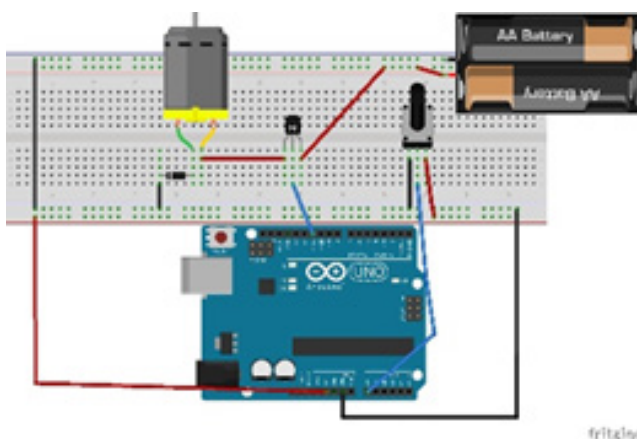
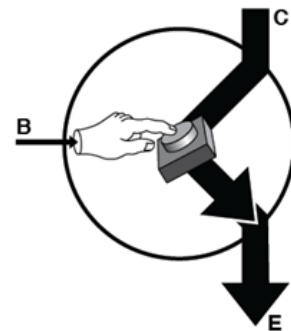
Bij erg grote stroomsterktes moet je een grotere transistor gebruiken of eentje met een koelelement zodat de transistor z'n warmte kwijt kan.



Opdracht 14 De snelheid van een elektromotor deel 2

De schakeling van opdracht 13 is te eenvoudig en het kan goed zijn dat de Arduino te weinig stroom levert. Zeker wanneer je de DC motor snel wilt laten draaien.

- Bouw de schakeling zoals hieronder weergegeven.
- Gebruik het script van opdracht 12 en upload het programma. Kan de elektromotor daadwerkelijk harder draaien?



Programmeren deel 4

Voor het aansturen van sommige apparaten zijn al scripts geschreven waardoor het aansturen van de apparaten eenvoudiger gaat. Een van die apparaten is een servo motor. Een servo motor is een soort elektromotor die 180° kan draaien. De rotatiehoek kan nauwkeurig ingesteld worden doordat deze werkt met een ingebouwde potmeter. Om gebruik te maken van een servo hebben we het volgende script nodig.

```
#include <Servo.h>
Servo mijnServo;
int pos = 0;
void setup() {
  mijnServo.attach(9)
}
```



Eerst wordt het script uit de bibliotheek opgehaald. Daarna noemen we de servo mijnServo en krijgt deze de begin positie 0.

Met mijnServo.attach(9) zeggen we dat de servo verbonden is met pin 9. Uiteraard is dit weer een PWM poort.

Nu is het mogelijk de servo te sturen. Waarbij pos staat voor de positie, die kan tussen 0 en 180 zitten. De code voor het sturen van deze servo wordt dan: mijnServo.write(pos); Een servo motor kan niet helemaal rond draaien zoals een gewone elektromotor maar we kunnen wel heel goed richten met een servomotor. Dit komt omdat er in de Servo een potmeter zit, de sturing gaat dus via het uitlezen van de spanning!

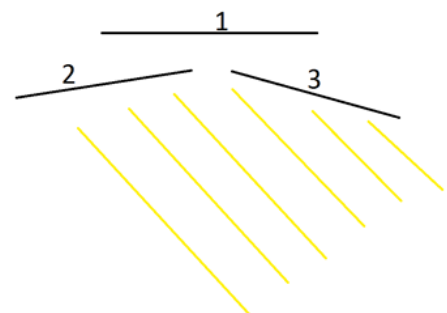
Opdracht 15 Een lichtwijzer

Gebruik opdracht 8 als basis. Vervang daarin de LED door de servo motor. Je kunt op de servo een wijzer plakken en een schaalverdeling maken zodat de wijzer aangeeft hoe licht/donker het is.

Opdracht 16 Zo veel mogelijk zonlicht

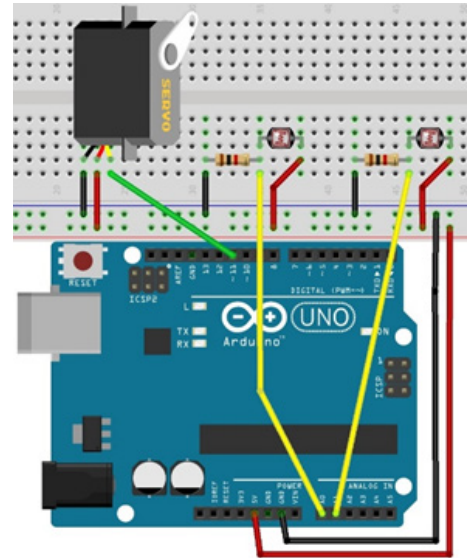
Zonnepanelen worden neer gezet om stralingsenergie, afkomstig van de zon, om te zetten in elektrische energie. De panelen vangen het meeste licht op als ze loodrecht op de zon staan. De aarde draait en we zouden de panelen mee moeten laten draaien om zoveel mogelijk energie op te vangen. Dat gaan we in deze opdracht doen.

Op een groot veld staan alle zonnepanelen opgesteld zoals zonnepaneel 1. Zonnepaneel 2 en 3 staan beide een beetje gedraaid ten opzichte van paneel 1. Je kunt nu goed zien dat de zonnepanelen de meeste elektrische energie produceren als zonnepaneel 2 en 3 even veel elektrische energie produceren. Paneel 2 vangt nu meer lichtstralen op dan paneel 3. Het hele systeem zou dus linksom gedraaid moeten worden.



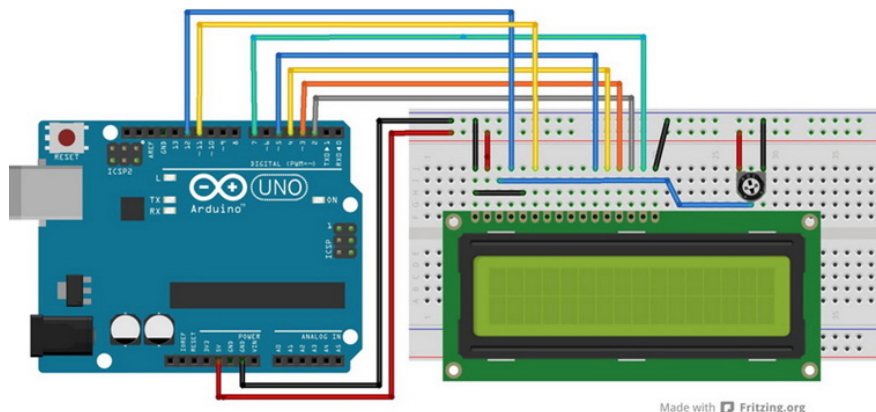
In plaats van een zonnecel beginnen we met een LDR.

- Bouw de schakeling en programmeer het eerste stuk waarbij je de waarden van de LDR kan uitlezen.
- Zet de flitser van je telefoon aan en lees met behulp van `Serial.println(LDR1)`; de waarden van LDR 1 en LDR2 uit. Beweeg je telefoon van links naar rechts en bekijk hoe de waarden variëren.
- Schrijf nu het stukje code voor de servo. Laat de servo naar links draaien als LDR1 meer licht ontvangt dan LDR2 en andersom. Denk eraan dat de servo gebruik maakt van een potmeter en je dus ook gebruik kan maken van de map functie.
- Leg uit dat al aardig in de buurt komt van het draaien van de zonnepanelen.



Opdracht 17 LCD

Met een LCD scherm (liquid cristal display) is het mogelijk om informatie te printen op een klein scherm. Daardoor is het niet meer nodig om gebruik te maken van de seriële monitor (en de computer) om gegevens uit te lezen.



- Maak de opstelling die hier staat.
- Open het programma HelloWorld, deze staat bij File/examples/LiquidCrystal en upload het programma. Let op! Bij HelloWorld staat je backlight nog niet aan, deze stuur je aan met pin 7 (zet het stukje code er in). Je kunt ook de backlight altijd aan hebben, verbind dan niet met pin 7 maar met de +.
- Probeer het script te lezen en kijk of je snapt wat de LCD laat zien.

Het script maakt gebruik van een library. Een library is een stuk code dat geschreven is door anderen, hierdoor wordt het programmeren je een stuk makkelijker gemaakt. Van heel veel verschillende sensoren zijn al libraries gemaakt, het is dus alleen zaak deze te vinden. Vaak lukt dat wel via bijvoorbeeld instructables.com of arduino.cc. De mogelijke functies staan dan aangegeven. Alle mogelijke functies voor deze library staan op: <https://www.arduino.cc/en/Reference/LiquidCrystal>

Even kijken naar het script: Na het inlezen van de library wijzen we de pinnen aan die gebruikt worden. In de setup wordt aangegeven dat de lcd 16 kolommen en twee rijen heeft (`lcd.begin(16,2);`). Ook wordt er meteen de tekst hello, world geprint (alles in de setup wordt 1x gedaan!).

In de loop wordt de tijd dat de Arduino aan is (`millis();`) geprint naar de LCD, en wel naar regel 2.

- Breid je schakeling uit met een drukknop en verander de code zodat het lcd scherm laat zien hoe vaak je op de knop hebt gedrukt.

Functies

Er is een gigantische lijst met functies. Hier de veelgebruikte functies in Arduino.

Functie	Uitleg
<code>int</code>	waarde tussen -32768 en 32767 (215)
<code>long</code>	waarde tussen -2.147.483.648 en 2.147.483.647 (231)
<code>unsigned long</code> <code>unsigned int</code>	/ waardes zijn alleen positief
<code>char</code>	tekens opgeslagen via het ASCII systeem
<code>float</code>	zijn decimale waardes, maar wordt zoveel mogelijk omgezet in Arduino programmeertaal
<code>digitalWrite(pin, waarde);</code>	schrijft geen spanning (LOW) of een spanning van 5 V naar de pin.
<code>analogWrite(pin, waarde)</code>	schrijft met een 8 bit output waardes tussen 0 en 5 V. De aangegeven waarde moet tussen 0 en 255 liggen.
<code>digitalRead(pin)</code>	leest of er een spanning op de pin staat, geeft een 0 of 1 terug.
<code>analogRead(pin)</code>	leest hoeveel spanning op de pin staat, geeft een waarde tussen 0 en 1023 terug.
<code>map(W,Min1,Max1,Min2,Max2)</code>	Is een schaalfunctie. Converteert een waarde (W) die ligt tussen Min1 en Max1 naar een waarde die ligt tussen Min2 en Max2.
<code>if(voorwaarde){}</code>	Als er voldaan wordt aan de gestelde voorwaarde, wordt de code tussen de accolades uitgevoerd.
<code>while(voorwaarde){}</code>	Zolang er aan de voorwaarde voldaan wordt, wordt de code tussen de accolades uitgevoerd.
<code>for(SW;EW;VF){}</code>	Een for-loop voert de code tussen de accolades een aantal keer uit. SW is de startwaarde, EW de eindwaarde en VF de verhogingsfactor. Vb: <code>for(int i = 0; i<20;i++){delay(100);}</code>
<code>switch case</code>	switch case is een soort keuzemenu waarbij een stukje code uitgevoerd kan worden. Als de variabele waarde een heeft dan voert deze de code uit behorende bij case 1. Zie onderstaand code blok.
<code>&&</code>	Beide voorwaardes moeten gelden
<code> </code>	Een van de twee voorwaardes moet gelden
<code>!=</code>	Is ongelijk aan.
<code>Tone(pin,f,t);</code>	Kan specifieke frequentie produceren. In de toon moet aangegeven worden op welke pin de luidspreker zit, welke frequentie gemaakt moet worden en eventueel hoe lang de toon aangehouden moet worden. Het is niet noodzakelijk aan te geven hoe lang de toon aangehouden moet worden.
<code>noTone(pin);</code>	Stopt de toon die geproduceerd werd.
<code>millis();</code>	Dit kan dienen als een timer. <code>millis();</code> geeft het aantal milliseconde dat de Arduino aan staat weer.
<code>micros();</code>	Geeft het aantal microseconde weer dat de Arduino aan staat. De resolutie is 4 µs.
<code>delay();</code>	Wacht een aantal milliseconde voor het uitvoeren van het volgende stukje code.
<code>random(LW,HW)</code>	Random geeft een getal terug tussen de laagste waarde (LW) en hoogste waarde (HW). Een nadeel van random is dat bij het opnieuw starten van de Arduino dezelfde sequentie wordt afgespeeld. Random is dus niet random...Dit probleem kan wel opgelost worden door eerst gebruik te maken van de functie <code>randomSeed(A0);</code> Deze leest pin A0 uit. Als daar niets is verbonden wordt hier ruis gemeten. <code>randomSeed</code> schudt als het ware de vaste sequentie van Random door elkaar.

```
switch (var) {
  case 1: \\ doe iets als var is 1
    break;
  case 2: \\ doe iets als var is 2
    break;
  default: \\ als geen van de bovenstaande waar
    is, doe dan het volgende (optioneel)
    break;
}
```

Attachment: code Blink.ino

```
/*  
  Blink  
  Turns on an LED on for one second, then off for one second, repeatedly.  
  
  Most Arduinos have an on-board LED you can control. On the Uno and  
  Leonardo, it is attached to digital pin 13. If you're unsure what  
  pin the on-board LED is connected to on your Arduino model, check  
  the documentation at http://www.arduino.cc  
  
  This example code is in the public domain.  
  
  modified 8 May 2014  
  by Scott Fitzgerald  
*/  
  
// the setup function runs once when you press reset or power the board  
void setup() {  
  // initialize digital pin 13 as an output.  
  pinMode(13, OUTPUT);  
}  
  
// the loop function runs over and over again forever  
void loop() {  
  digitalWrite(13, HIGH);    // turn the LED on (HIGH is the voltage level)  
  delay(1000);               // wait for a second  
  digitalWrite(13, LOW);     // turn the LED off by making the voltage LOW  
  delay(1000);               // wait for a second  
}
```

Attachment: code Fade.ino

```
/*
  Fade

  This example shows how to fade an LED on pin 9
  using the analogWrite() function.

  The analogWrite() function uses PWM, so if
  you want to change the pin you're using, be
  sure to use another PWM capable pin. On most
  Arduino, the PWM pins are identified with
  a "~" sign, like ~3, ~5, ~6, ~9, ~10 and ~11.

  This example code is in the public domain.
  */

int led = 9;           // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup() {
  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

Attachment: code Button.ino

```
/*
  Button

  Turns on and off a light emitting diode(LED) connected to digital
  pin 13, when pressing a pushbutton attached to pin 2.

  The circuit:
  * LED attached from pin 13 to ground
  * pushbutton attached to pin 2 from +5V
  * 10K resistor attached to pin 2 from ground

  * Note: on most Arduinos there is already an LED on the board
  attached to pin 13.

  created 2005
  by DojoDave <http://www.0j0.org>
  modified 30 Aug 2011
  by Tom Igoe

  This example code is in the public domain.

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

// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin = 13;      // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```


Attachment: code AnalogReadSerial.ino

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Graphical representation is available using serial plotter (Tools > Serial
  Plotter menu)
  Attach the center pin of a potentiometer to pin A0, and the outside pins to
  +5V and ground.

  This example code is in the public domain.
*/

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```