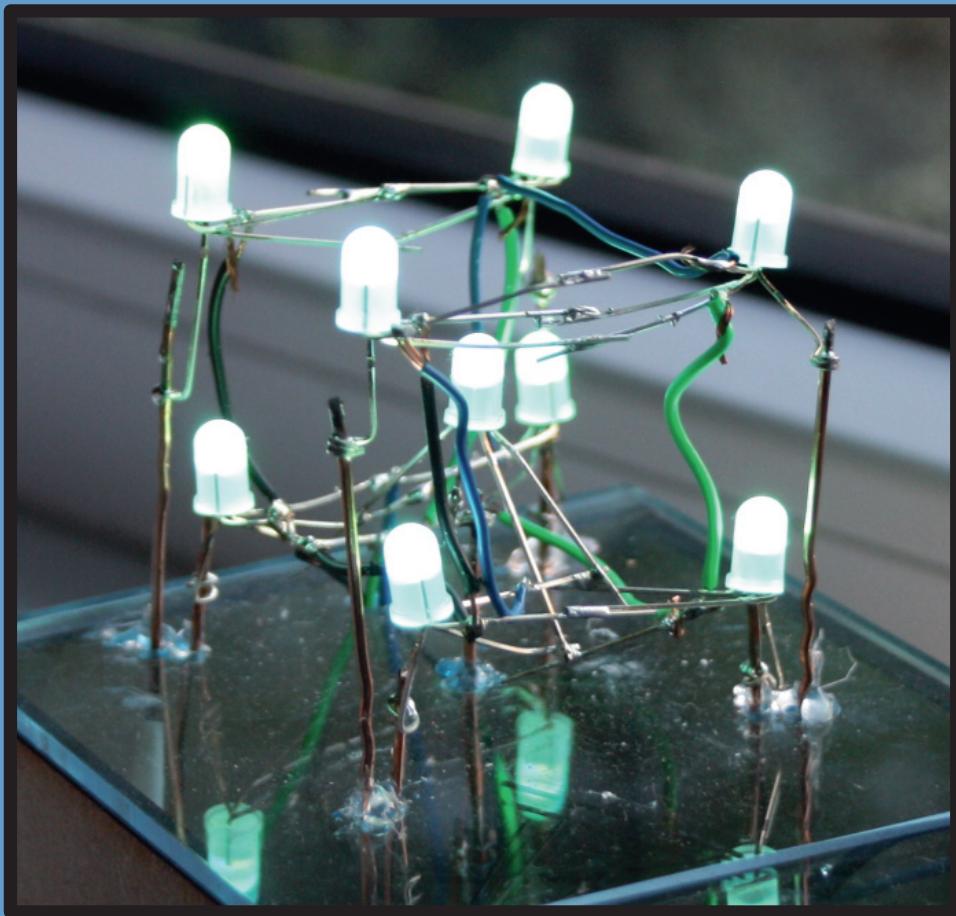


Fe-Cube

A Maker's Guide Handleiding



Benny Malengier
M.Cristina Ciocci

De Fe-Cube kit

Proficiat met je aankoop van de Ingegno Fe-Cube kit. Laat je fantasie de vrije loop, en maak je eigen veelkleurig licht.

Je dient volgend gereedschap klaar te leggen om alles ineen te kunnen zetten:

- Soldeerstation en soldeertin
- Secondenlijm of lijmpistool
- Striptang, Kniptang
- Een vrije werkruimte



Met deze kit leer je werken met Arduino, solderen je componenten en bouw je een hoop persoonlijk licht.

Bij deze kit hoort Arduino code die je kunt downloaden van <https://github.com/ingegno/fecube>

Principe

Een kleuren LED (RGB LED) heeft 4 pootjes. We hebben 3 signalen nodig om hem te sturen. We zullen 9 RGB LED sturen in dit project, dus $3 \times 9 = 27$ signalen zijn nodig. Daarnaast willen we ons licht kunnen sturen met een drukknop (een extra signaal), alsook optioneel een afstandssensor inbouwen (3 signalen voor nodig).

Om dit stukje electronica mogelijk te maken hebben we dus minstens 25 signaalkanalen nodig. Dat is bijgevolg al een ingewikkeld schema wat met een eenvoudige rekeneenheid (de Arduino UNO) al niet meer haalbaar is.

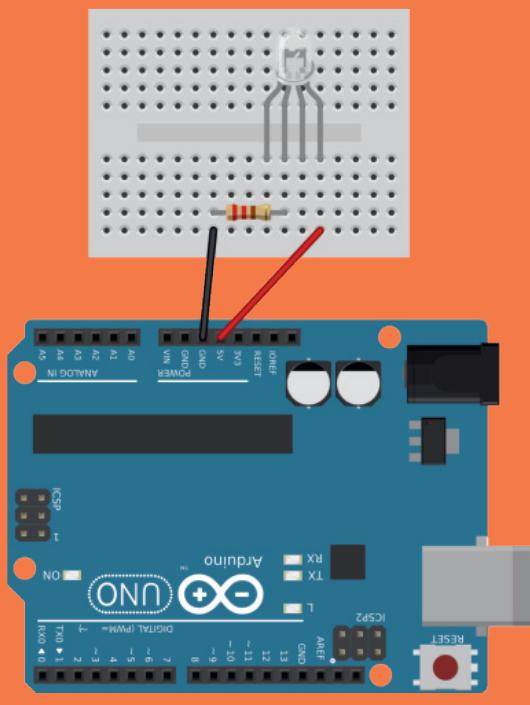
We zullen bijgevolg in dit project gebruik maken van een vereenvoudiging om ons licht mogelijk te maken: *multiplexen*. Dit houdt in dat we een groep lichten controlen als een product van twee signaal groepen. We nemen een groep met 3 kleuren, een groep met 9 LED. Hiervoor zijn dan 12 signaalkanalen nodig, een grote reductie van onze oorspronkelijke 21 signalen voor de LED.

De Componenten

- Lasercut frame om de electronica in te stoppen
- Een plexiglas bovenlaag
- Tien RGB LED (Common Cathode)
- Tien 220 Ohm of 330 Ohm weerstanden
- Een 10k Ohm weerstand (of groter)
- Een drukknop
- Een Arduino UNO met USB kabel
- 5 harde draden die we als pilaren voor de kubus zullen gebruiken (paperclips)
- 3 NPN transistoren (2N3904 of 2N2222 of BC547)
- Een schakelbord (breadboard)
- Verschillende draadjes: rode, groene en blauwe draad voor kleur circuits, en ongeveer 24 connectordraadjes voor de Arduino en het schakelbord.
- 2 M3 schroeven en bouten
- 9 kleine plastieken cilinders

Om de LEDs te sturen gebruiken we een Arduino microcontroller. Dit is een processor die we kunnen programmeren om lichteffecten te maken. Je kan in dit project de Fe-Cube bouwen en effecten programmeren, of je beperken tot het bouwen en de voorgemaakte effecten gebruiken. In elk geval dien je de Arduino software te installeren en leren hoe je het bord programmeert.

Eerste schema



Arduino

Laat ons beginnen met onze Arduino uit te testen. Neem je schakelbord, je Arduino, 3 draden, een weerstand en een RGB LED. De RGB LED zijn Common Kathode LED. Dit betekend dat het langste been negatief moet zijn, en de andere beentjes positief om licht te genereren.

Laat ons dit uittesten met onze Arduino UNO. Neem de Arduino, een 220 Ohm weerstand en een RGB LED. Het langste been is de kathode en dient verbonden te worden met de GND van de Arduino via de weerstand. Een van de andere benen verbind je met de 5V input. Normaal is de orde van de kleuren op je RGB LED links rood, dan de kathode, dan blauw, en rechts groen.

Zet je Arduino onder stroom door de usb kabel met je computer te verbinden. Test via dit schema of je LEDs werken.

Dat was simpel niet? Geen programmatie was nodig, we gebruikten de Arduino hier gewoon als een batterij. De Arduino hebben we evenwel omdat we het licht willen controleren via code. Laat ons dit doen en de RGB laten flikkeren met enkele kleuren.

Ga met je webbrowser naar <http://arduino.cc>, en klik op de Download button. Download dan de Arduino IDE. Kies de stabiele versie. Laat je hierbij helpen door een volwassene.

We zullen nu ons licht de 3 kleuren laten tonen. Eerst wijzigen we ons schema. In plaats van de 5V output te

gebruiken, gebruiken we nu de genummerde Arduino uitgangen. Zie het schema hiernaast. We hebben de pootjes van de LED die overeenkomen met de kleuren verbonden met de uitgangen 8, 9 en 10 op de Arduino.

Als tweede stap dienen we nu een programma te schrijven met de Arduino IDE die zorgt dat deze poorten stroom krijgen en de LED doen branden. Open dus de Arduino IDE (laat een volwassene controleren dat in menu Extra->Bord het juiste bord aangeduid is, normaal zou je de Arduino Uno moeten hebben), en tik volgende code in:

```
const int ledPinR = 8;
const int ledPinG = 9;
const int ledPinB = 10;

int ledStateR = LOW;
int ledStateG = LOW;
int ledStateB = LOW;
long previousMillis = 0;

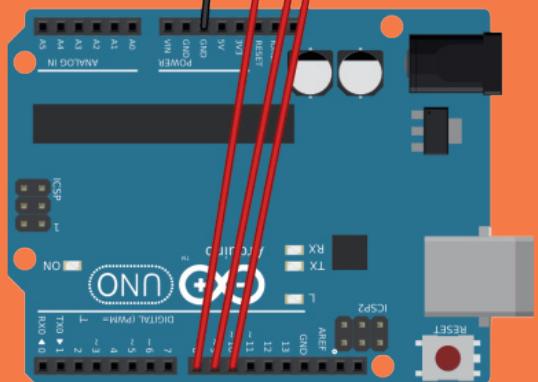
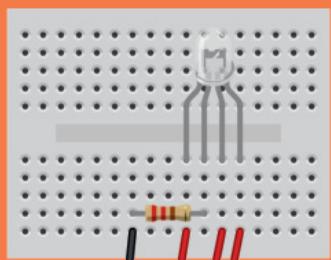
long interval = 3000;

void setup() {
    pinMode(ledPinR, OUTPUT);
    pinMode(ledPinG, OUTPUT);
    pinMode(ledPinB, OUTPUT);
}

void loop()
{
    unsigned long currentMillis = millis();
    unsigned long timer = currentMillis
                          - previousMillis;
    if (timer > interval) {
        // save last time
        previousMillis = currentMillis;
    }

    if ( timer < interval/3) {
        ledStateR = HIGH;
        ledStateG = LOW;
        ledStateB = LOW;
    } else if (timer < 2*interval/3) {
```

RGB LED besturen



Basis Arduino Code

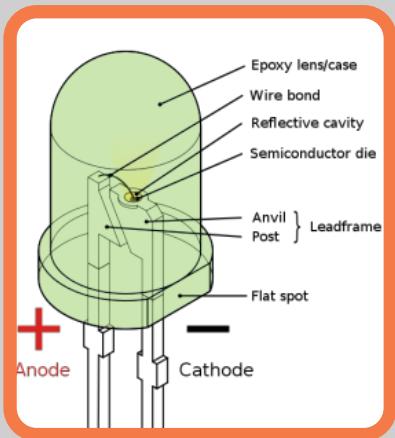
- ; Elk opdracht eindigt met een puntkomma
- { Accolades gebruik je voor een blok code
- // Twee slashes start commentaar
Dit is dus geen code. Gebruik het om je code leesbaar te maken!
- /* commentaar */ is een alternatieve methode voor lang commentaar

if (conditie) { } else { ...}

De if conditie laat je toe te testen. Is de conditie waar, dan voer je eerste codeblok uit. Staat er een else, dan wordt dat blok uitgevoerd indien de conditie vals was.

== Testen of iets gelijk is
<, **<=**, **>**, **>=** Testen of iets kleiner, kleiner of gelijk, groter of gelijk, groter is.

Meer: http://ingegno.be/Manuals/Ard_Sheet_nl.pdf



```

ledStateR = LOW;
ledStateG = HIGH;
ledStateB = LOW;
} else {
    ledStateR = LOW;
    ledStateG = LOW;
    ledStateB = HIGH;
}
// set the LED
digitalWrite(ledPinR, ledStateR);
digitalWrite(ledPinG, ledStateG);
digitalWrite(ledPinB, ledStateB);
}

```

Bovenstaande code moet identiek overgetikt worden in de Arduino IDE. Een programmeertaal moet verstaan worden door een computer. Spijtig genoeg zijn computers nog altijd een beetje dom, dus mag je geen enkele fout maken! Het is zoals een dictee waar je altijd 10/10 moet halen. Neem dus de tijd om de structuur en taal van de Arduino te leren. Zie de blaadjes 'Programmeren met Arduino' die je bij deze gids kan downloaden.

Ben je klaar? Druk dan op de eerste knop in de Arduino IDE.



Onderaan de IDE zul je of zien dat Compileren voltooid is, of zul je een fout krijgen. De fouten kunnen moeilijk te lezen zijn. Laat je bijstaan door een volwassene indien nodig. Een foutmelding bevat de lijn waar een fout gezien is. Bevoorbeeld:

Fout bij compileren.

```

n01_RGB.ino: In function 'void loop()':
n01_RGB.ino:26:3: error: expected ';' before ')' token
n01_RGB.ino:29:17: error: 'HIGHH' was not declared in this scope

```

25

Hier zie je dat er een fout is op lijn 26. Een ; teken ontbreekt. Dit is een typische fout die je snel kunt corrigeren. De tweede fout is op lijn 29. HIGHH werd ingetikt ipv HIGH. De lijn waar je cursor op staat zie je in de onderaan de ide. In de figuur hierboven zie je dat dat lijn 25 was.

Heb je een fout? Niet panikeren, lees de foutbericht rustig door, en probeer te begrijpen wat er gaande is.

Als er geen fouten meer zijn kun je de code opladen naar de Arduino met de laadknop:

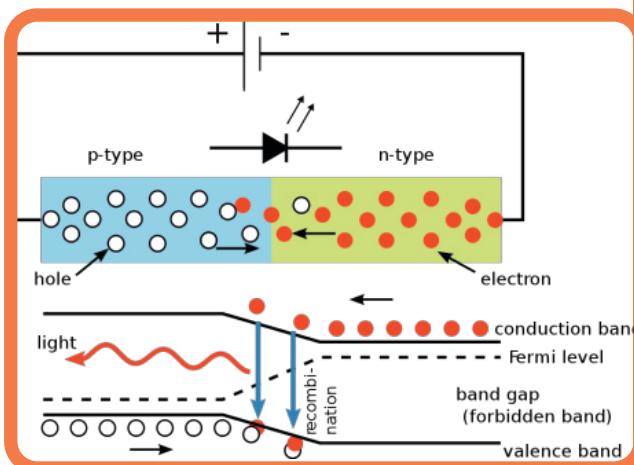


Via de USB kabel wordt de code op je Arduino geladen, en als je het schema juist hebt zul je de RGB LED zien beginnen flikkeren. Proficiat, je eerste Arduino sketch is af!

Problemen met je code? Je kan ze ook downloaden van volgende webpagina:

<https://github.com/in gegno/fecube>

Ga naar de sectie **Manual sketches**, en selecteer de Arduino sketch van deze handleiding die je wil downloaden of copiëren. Voor deze eerste sketch was dat **n01_RGB.ino**.



Code uitgelicht

Je code werkte en de RGB LED flikkert. Maar hoe werte ons programma? Laat ons de verschillende lijnen bekijken en uitleggen.

De Arduino heeft genummerde uitgangen. We gebruiken uitgang 8, 9 en 10. Daarom begint ons programma met het definieren van namen voor deze nummers, om later geen fouten te maken. Zo zullen we de pin op de LED voor rood verbinden met uitgang 8. Daarom schrijven we

```
const int ledPinR = 8;
```

De 8 is een constante (const), en een geheel getal (integer in het Engels, daarom int). Telkens als we later ledPinR schrijven, bedoelen we het nummer 8.

Hoe werkt een LED?

We beschrijven dit met kwantumfysica, een tak van de fysica waar de meeste mensen nooit over leren. We werken dan met kwantumdeeltjes, hier elektronen, negatief geladen deeltjes. Elektronen stromen uit de GND pin, passeren de kathode (-) van de LED, en dan de anode (+).

De cathode bestaat uit wat we een n-type materiaal noemen. Dit is een materiaal dat een overschat aan elektronen heeft. De anode zijde bestaat uit een p-type materiaal. Dat is een materiaal dat een tekort aan elektronen heeft, wat we gaten noemen: het is de afwezigheid van een elektron in het materiaal. Het p-type materiaal geleidt stroom door gaten rond te bewegen. Het n-type materiaal via elektronen.

Dus, als elektronen toekomen aan de cathode creëren ze daar een surplus aan elektronen, en de elektronen worden geduwd in de richting van het p-materiaal. Daar aangekomen vallen ze in een van de gaten, en het is dit kwantumeffect dat een klein kwantum (beetje) aan energie creëert: een foton, wat we waarnemen als het licht van de LED. Dit effect noemen we Electroluminescentie, en het gebeurt enkel bij overgang van een elektron van een n-type materiaal naar een p-type materiaal waarbij een elektron in een gat valt.

Een LED zal dus maar licht produceren met stroom die in een specifieke richting vloeit: kathode verbonden met GND! Probeer maar eens andersom.

Opdracht 1

Probeer nu de code te veranderen. Je kan het volgende proberen:

- Wijzig hoe lang de verschillende kleuren getoond worden. Nu word elke kleur 1/3 van de intervallengte getoond. Toon rood langer, of maak een eigen afwisseling van RGB kleuren.
- Wat gebeurt er als je de intervallengte korter en korter maakt? Het kortst mogelijke interval met onze code is 1 milliseconde, of een duizendste van een seconde. Je kan nog vlugger met Arduino via de functie `micros()`, welke in microseconden rekent, een miljoenste van een seconde.
- Wat stel je vast als je terzelfderijd Blauw en Groen, of zelfs alledrie, stroom heeft? Dit kan door HIGH te sturen naar de juiste pinnen via de `digitalWrite` functie.

De Weerstand

Waarom was er een weerstand nodig in ons schema om de LED te doen branden?

De Arduino levert 5 Volt. Volt is een indicatie van hoeveel werk je kan doen. De LED is gemaakt voor 3 Volt. Gebruik je meer dan zal hij kapot gaan.

We moeten daarom een weerstand gebruiken om het voltage over de LED te verminderen. De resistor of weerstand doet wat de naam zegt: het resisteert of weerstaat de stroom. Bijgevolg verliezen de kwantumdeeltjes een deel van hun kracht als ze door een weerstand stromen. Niet alleen zullen de deeltjes minder kracht hebben erna, er zullen ook minder deeltjes passeren elke seconde. De stroom is dus verminderd. Een weerstand of resistor zal dus de stroom verminderen en na het passeren zal er ook nog eens minder kracht over zijn om door de rest van het circuit te stromen.

De uitgangen van de Arduino kunnen twee voltages hebben: GND (grond, 0V), of 5V. We duiden deze aan met LOW of HIGH. We definieren met

```
int ledStateR = LOW;
```

een variabele die LOW of HIGH kan zijn.

Vervolgens definieren we ook een getal dat bijhoudt welke ons laatste tijdstip was waarbij we de LED een flikkercyclus lieten beginnen. We noemen dit previousMillis. De long ervoor betekent dat dit een heel lang geheel getal kan zijn.

Vervolgens houden we bij met interval hoe vlug we willen flikkeren. Dit is in milliseconden, dus 3000 betekent dat een cyclus 3 seconden duurt.

Elk Arduino programma dient een setup en een loop deel te hebben. De setup wordt aan het begin uitgevoerd, de loop is een deel dat constant herhaald wordt. Het enige wat we de Arduino moeten meedelen bij opstarten is welke uitgangen zullen gebruikt worden en hoe ze gebruikt worden. In ons geval worden ze gebruikt om iets stroom te geven, wat aangeduid wordt met OUTPUT. De uitgang voor het rode LEDbeen als uitgang zetten is dus de lijn

```
pinMode(ledPinR, OUTPUT);
```

De loop is ons programma. We zullen een derde van ons interval rood tonen, dan groen, dan blauw, en dan herbegint de cyclus. Hiertoe houden we een timer bij, die aanduidt welk tijdstip het nu is. Via de functie `millis()` bekomen we de tijd sinds de Arduino aangezet werd, de timer is dus verschil van het huidige tijdstip en het begin van een cyclus.

```
unsigned long timer = currentMillis - previousMillis;
```

Als de timer groter wordt dan het flikkerinterval, starten we weer bij het begin door ervoor te zorgen dat timer de volgende keer 0 zal worden.

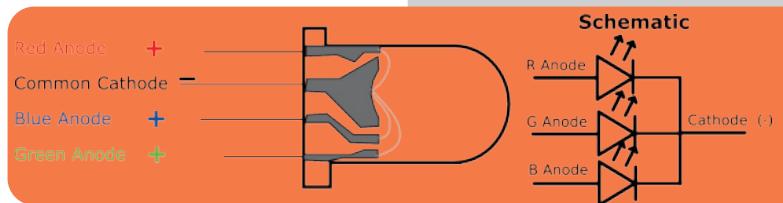
```
previousMillis = currentMillis;
```

Hierna bepalen we welke kleur we zullen tonen

op basis van de waarde van timer. Het eerste deel rood, dan groen, dan blauw. We eindigen met naar de uitgang van de Arduino de berekende stroom te sturen:

```
digitalWrite(ledPinR, ledStateR);
```

Ons programma bevat dus: bovenaan definieren van nodige getallen, daarna in setup definieren van welke uitgangen van de Arduino nodig zijn, en eindigen met in loop te berekenen welk beentje stroom moet krijgen en welk niet, en via digitalWrite er dan voor te zorgen dat dat gebeurt.



De Transistor: een mini schakelaar

We hebben de LED doen branden met een uitgang van Arduino. Voor een LED is dat geen probleem, maar meer algemeen lever je beter stroom aan componenten via de 5V en GND uitgangen van de Arduino. Indien we dat doen, zou de LED altijd branden, gezien er geen manier is om de stroom op de 5V of GND uitgang af te schakelen.

Daarom dienen we een schakelaar tussen de stroom te plaatsen, welke we open of toe kunnen zetten. Een transistor is een mini-schakelaar die dat voor ons kan doen.

Er bestaan veel soorten transistoren. We zullen de transistor NPN 2N3904 of 2N2222 of BC547 gebruiken. Wat doen de beentjes? Hou je de transistor rechtop en kijk je naar het vlakke stuk van de NPN, dan is voor de BC547 transistoren die wij gebruiken de linkerpoort de in-zijde voor de stroom (C, van collector, waar er gecollecteerd wordt), de rechterpoort de uitgangzijde (E, van emitter of extruder) die naar de GND gaat, en het middelste been is de P of positieve controle (aangeduid met B, van basis). Als er een positief voltage (HIGH) is op de basis, laat de NPN transistor stroom door. Is de basis verbonden met de grondstroom (LOW), dan kan er geen stroom door en is het circuit dus onderbroken. De 2N3904 of 2N2222 hebben de collector en emitter precies andersom.

Maak met je componenten nu de schakeling zoals je ziet in

Hoe werkt een Transistor?

Er bestaan veel type transistoren. Je vindt ze in alle maten en vormen. Er zijn ook verschillende manieren om ze te maken. Zo bestaat de chip in je computer ook uit miljoenen superkleine transistoren.

De transistor die wij gebruiken is van het type Bipolaire transistor. In het stuk "Hoe werkt een LED" hebben we gezien dat een LED een PN overgang is. We noemen onze transistor een NPN transistor omdat hij inderdaad een combinatie is van 3 materialen die geconnecteerd worden: een n-type materiaal, een p-type materiaal, en dan opnieuw een n-type materiaal. Deze keer heeft het p-type materiaal maar weinig gaten. Als we het verbinden met de GND (-) zullen er nog minder gaten zijn. Zoals we uitgelegd hebben, een p-type materiaal geleidt stroom via de gaten, dus kan er geen stroom vloeien als alle gaten gevuld zijn, en is de schakelaar open.

Als we een voltage over het p-type materiaal plaatsen, verliest het materiaal zijn elektronen, en krijgt dus meer gaten. Dit betekent dat we nu stroom kunnen hebben van het p-materiaal welke we de basis noemen, naar de emitter, welke een van de twee n-materialen is. Dit is hetzelfde als de PN overgang in de LED. De emitter zendt elektronen in de basis (emit is Engels voor uitzenden). Let op, bij conventie stroomt de stroom van plus naar min, dus het tegengestelde van wat hier echt gebeurd: de elektronen bewegen van emitter naar basis, en de stroom van basis naar emitter. In het n-type materiaal stromen de gaten zoals de stroom.

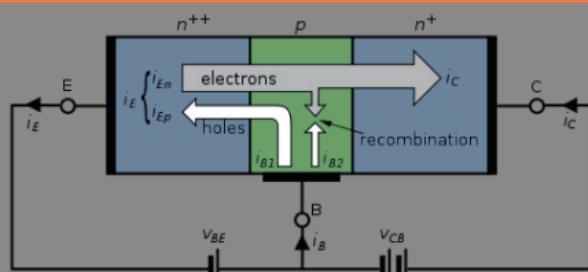


Fig: Een NPN transistor

Goed, hoe krijgen we dan stroom van het n-type materiaal naar het andere n-type materiaal? Dat zou niet mogelijk mogen zijn! De truck is dat het p-type materiaal eenmaal verbonden met een voltage (+) extra gaten krijgt, en het kan een lijn van gaten vormen van het emitter n-type materiaal naar het collector n-type materiaal. Nu gebeurt iets speciaals: de elektronen kunnen springen van gat naar gat via deze lijn van de emitter naar de collector. En wat meer is, ze kunnen dat veel, veel, vlugger doen dan dat de gaten zelf kunnen bewegen. In andere woorden, we krijgen een versterking van de stroom aan de basis, zie de figuur, de schakelaar is gesloten. Als we de stroom aan de basis verhogen, stromen er nog meer gaten in het p-type materiaal, en kunnen er meer 'verbindingsslijnen' tussen emitter en collector. We kunnen dus een klein signaal aan de basis omvormen tot een groot signaal over de NPN.

Merk op dat we in ons circuit een 5V signaal over de basis plaatsen via de Arduino uitgang, welke op zijn eentje al genoeg is om de LED te doen branden. Hierdoor is onze stroom aan de basis al genoeg om de LED aan te sturen. Dit is geen probleem, gewoon iets waar je bewust van moet zijn. Via de weerstanden naar de basis verlagen we deze stroom. Een weerstand 10 keer groter zou nog altijd werken in ons schema.

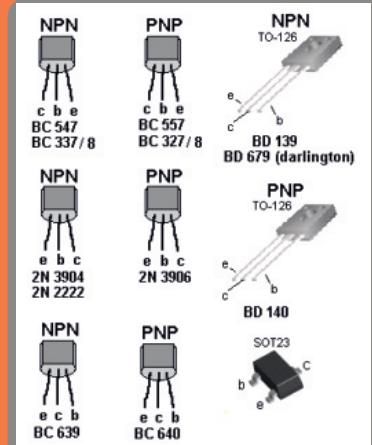
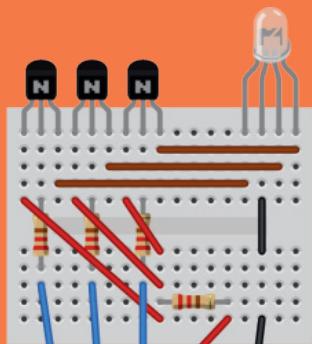


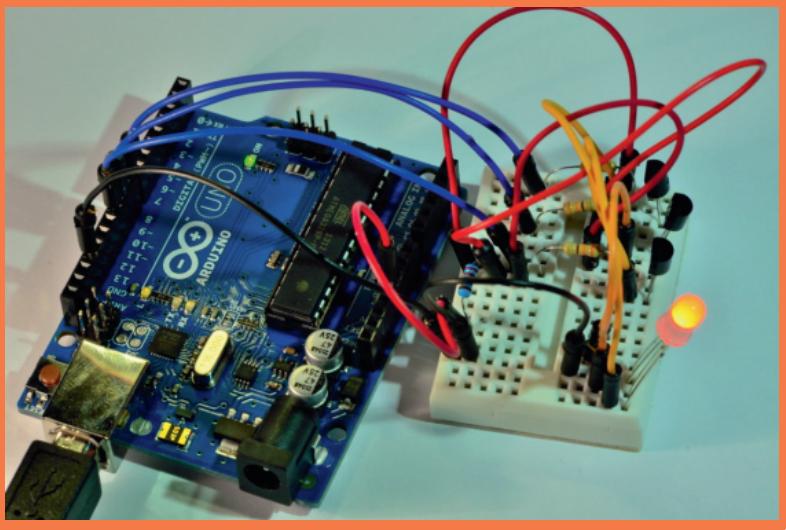
Fig: Verschillende type transistoren

de zijbalk "LED met NPN besturen". Je ziet dat je 3 NPN transistoren nodig hebt, 4 weerstanden, de RGB LED en veel draadjes. Als je klaar bent kun je je Arduino aanschakelen. Het programma om deze schakeling te doen werken is identiek aan het vorige programma. Met uitgang 8, 9 en 10 controleren we de drie kleuren. Het enige verschil is dat we nu niet rechtstreeks de LED doen branden, maar in plaats daarvan de schakelaar open of toe zetten.

LED met NPN besturen



Het Resultaat



Opdracht 2

Je hebt de resistoren gebruikt in de schakeling uit de LED kubus kit. Deze zijn evenwel niet zo sterk. De ene grote weerstand zou dan weer te sterk moeten zijn. Heb je van andere projecten weerstanden liggen? Test dan hoe groot de weerstand kan zijn.

Je zou in staat moeten zijn om de stroom over de Basis van de NPN sterk te verminderen op deze manier en toch een werkend circuit overhouden.

Zou dit een manier kunnen zijn om de sterkte van de resistor te testen?

Kleuren mengen

Persistentie van visie

Als je de opdracht uit de eerste sectie gedaan hebt, weet je dat door vlug te flikkeren, je de illusie kunt creeren dat de LED altijd helemaal niet flikkert maar een 'gemengde' kleur toont. Dus, we kunnen beïnvloeden hoe helder een LED is of welke kleur hij toont door heel snel andere kleuren te tonen, of hem even af te zetten (dimmen)

De truck is te verhinderen dat de mensen de LED effectief zien blinken. Gelukkig hebben mensen persistentie van beelden, denk aan de Phenakistiscoop, uitgevonden door de Belg Plateau in Gent. Persistentie van beelden is het fenomeen waarbij een beeld dat maar een fractie van een seconde gezien is, door je brein zal blijven gezien worden, zelfs nadat het originele beeld verdwenen is of bewogen heeft. Dit is het principe achter film en televisie.

Mensen kunnen maximaal 24 beelden per seconde zien, als we dus vlugger flikkeren dan ongeveer 40 milliseconden, zullen onze hersenen een samengesteld beeld waarnemen.

Als je opdracht 1 hebt gedaan, weet je dat je niet zomaar de rode, groene en blauwe kleur kunt mengen. Hoe kunnen we dan roze, of paars, bekomen via de RGB LED? Of kan dat niet?

Ja, dit kan! Dankzij persistentie van visie kunnen we onze ogen doen geloven elke combinatie van rood, groen en blauw licht te zien. Welke kleuren je zo kan maken? Een heleboel:



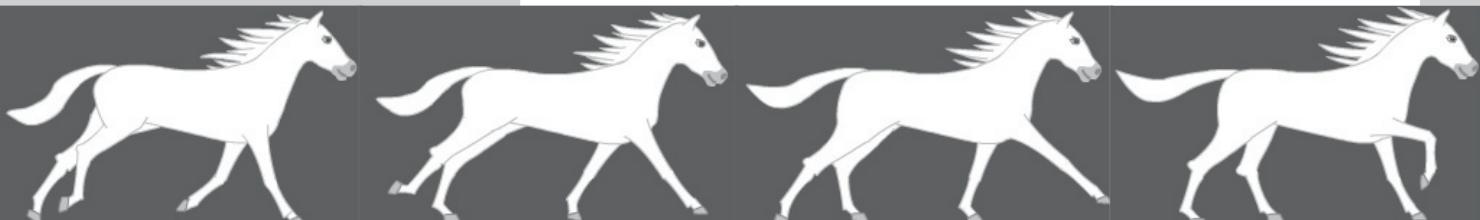
Laat ons een Arduino programma maken om een gemengde kleur te tonen op onze RGB LED. Het Arduino circuit wijzigt niet. We zullen enkel het programma 'slimmer' maken, en zo de gemengde kleuren tevoorschijn toveren.

Download het tweede programma van de website, [n02_RGB_mixing.ino](#), en sla het op in een map met dezelfde naam: **n02_RGB_mixing**. Open de Arduino sketch met de Arduino IDE, laad het op.

Je zou nu in plaats van rood, groen of blauw die flikkeren, drie andere gemengde kleuren moeten zien.

Laat ons kijken wat we verandert hebben in de sketch. Het eerste belangrijke is dat we extra functies definieren: de functie `color()` en de functie `yellow()` (geel in het Engels). Dit maakt de code overzichtelijker, en laat doe een stukje code die verschillende keren terugkomt maar een keer op te schrijven.

We mengen kleuren door een hoeveelheid rood, groen en blauw te tonen. De manier dat die hoeveelheid



Tonen we heel vlug deze paardenbeelden na elkaar, dan zal het lijken alsof het paard aan het rennen is. Op dit principe van persistentie van visie is het principe van de film gebaseerd

doorgegeven wordt is een afspraak. Wij gebruiken de meest voorkomende: een kleurhoeveelheid geven we door via een getal van 0 tot 255. Dus rood 0 betekent geen rood, 255 volledig rood, en 200 dat we 200 delen van 255 aan rood tonen.

De functie `color(int R, int G, int B)`, is een functie die de 3 hoeveelheden doorkrijgt, en dit omzet in een gemengde kleur. De truck is om voor elke eenheid aan kleur 3 microseconden te tonen. Is rood dus 2, dan tonen we 6 microseconden rood, is rood 200, dan tonen we 600 microseconden rood. Een vaste duur een kleur tonen doen we met de functie

```
delayMicroseconds(val);
```

Dit is een functie die evenlang wacht als de waarde die je opheeft als val.

Door enkel rood te tonen, en daarna dit te doen voor enkel groen en enkel blauw, slagen we erin een gemengde kleur te krijgen.

Om de kleur te dimmen moeten we daarna nog een tijdje geen kleur tonen. Dit is niet zo mooi om zien als het niet volledig donker is. Daarom kun je zelf kiezen of je dimmen wil toelaten of niet. Als de variabele `ALLOWDIM` waar is (true in het Engels), dan dimmen we. Zet je de variabele `vals` (false in het Engels), zoals wij standaard doen, dan is dimmen niet toegestaan. Met de code die we hier geven zal dimmen vlug aanleiding geven tot flikkeren, je bent gewaarschuwd!

De functie `yellow()` toont how we de functie `color` kunnen gebruiken om vaste kleuren te definieren. In dit geval geel, welke evenveel rood als groen is.

De loop functie dient ook aangepast te worden ten opzichte van vorig programma. We dienen niet meer zelf op te geven of een arduino uitgang aan of af is, dat doet de `color` functie nu. We dienen enkel nog maar een effect te maken. In dit geval een seconde geel tonen, dan een kleur die wit benadert, dan paars.

Opdracht 3

Wijzig de Arduino sketch als volgt: Laat dimmen toe, en toon een rood licht dat in 10 stappen van helrood naar compleet gedimd rood wijzigt. Neem als intervallengte 10 seconden, dus een wijziging in de helderheid elke seconde.

Multiplexen: meer doen met minder

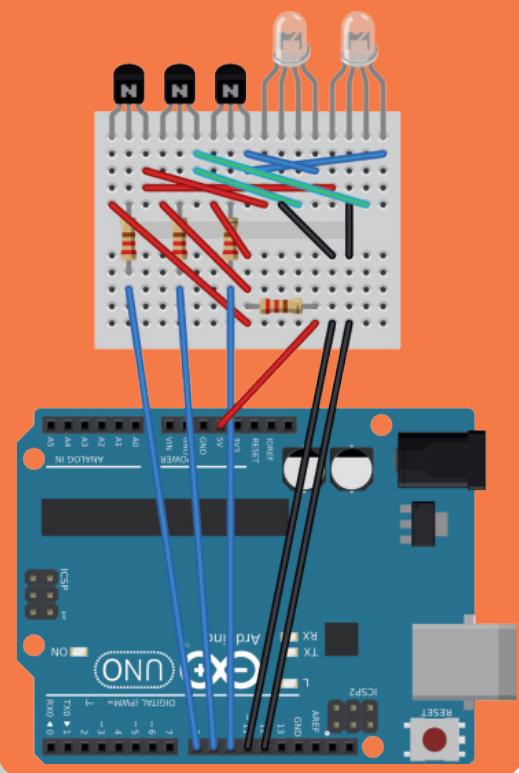
We hebben bijna alle principes uitgelegd waarom onze LED kubus gebaseerd is: de RGB LED, de transistor, persistentie van visie. Het laatste principe is multiplexen.

Multiplexen is het principe waarbij je met bv 6 uitgangen van de Arduino niet 6 LEDs controleert, maar wel bv $3 \times 3 = 9$ LEDs of $2 \times 4 = 8$ LEDs. We sturen dus meer LEDs aan dan we uitgangen hebben. De kostprijs die we moeten betalen is dat we niet alle LED terzelfdertijd kunnen doen branden, maar enkel maar een groep per keer. Door het principe van persistentie van visie is dit evenwel geen probleem, als we vlug genoeg schakelen tussen de groepen, zullen we toch een enkel beeld waarnemen.

Laat ons multiplexen aantonen met een schakeling die twee RGB LED bevat. We hebben dan 6 pinnen aan te sturen. In de vorige oefening hadden we pinnen 8, 9 en 10 nodig. Voor twee LED zouden we dus normaal nu pinnen 11, 12 en 13 nodig hebben. We kunnen dan beide LED terzelfdertijd sturen.

Als we nu multiplexing willen gebruiken zullen we de LED niet meer terzelfdertijd kunnen sturen, maar wel minder uitgangen nodig hebben. We weten dat $6 = 3 \times 2$, dus kunnen we met $3 + 2 = 5$ Arduino uitgangen de LEDs aansturen. Dit is maar een kleine winst aan uitgangen, maar voor meer sturing loopt de winst vlug op. Onze LED kubus zal 27 LED pinnen moeten sturen. We zullen dit doen met 12 pinnen in plaats van 27! Gezien de Arduino UNO maar 18 bruikbare uitgangen heeft, is multiplexen de enige manier om hem te doen werken.

LED met NPN besturen



Het circuit vind je in de zijkolom. Maak dit na. Het schema bouwt voort op het vorige: je voegt een RGB LED toe naast de eerdere RGB LED. Je verbindt de transistoren naar de R, G en B benen van beide LED. De common Kathode van de eerste LED verbind je met uitgang 11, en die van de tweede LED naar uitgang 12 in plaats van de GND waar het eerder zat. Indien er veel stroom over de LED zou stromen zouden we twee extra NPN transistoren toevoegen tussen de LED kathodes en de GND, en uitgang 11 en 12 gebruiken om die te sturen. Gezien een LED niet veel stroom nodig heeft doen we dit hier niet evenwel.

De Arduino sketch kun je downloaden van de website, het is het derde programma **n03_RGB_multiplex.ino**, welke je weer in een map **n03_RGB_multiplex** moet opslaan. We kunnen niet beide LED terzelfdertijd sturen, dus zullen we eerst de linkse LED, dan de rechtse LED sturen, en dit zo vlug dat het lijkt dat ze beide aan zijn: we multiplexen de twee LEDs.

De code is erg analoog aan de vorige Arduino sketch. De functie color is er nog steeds en is dezelfde. Een nieuwe functie LEDs() is aanwezig welke zorgt voor het multiplexen: eerst wordt de linkse LED kathode (-) op LOW, dus 0V gezet. De rechtse staat dan nog op HIGH (zie setup, dus op 5V). Bijgevolg brandt enkel de linkse LED. We roepen dan de color functie op om op die LED een kleur te tonen. Daarna wordt de linkse LED af gezet (op HIGH), en de rechtse aan (op LOW), en roepen we weer color op om op de rechtse de gewenste kleur te tonen. Dit alles gebeurt in een fractie van een milliseconde.

Het effect dat we standaard tonen in de loop() functie is iets anders: het duurt 2 seconden (interval = 2000), en de eerste seconde tonen we paars links, en geel rechts. De tweede seconde is dit omgewisseld. Daarna herbegint het effect.

Hierbij hebben we alle basisprincipes gezien van de LED kubus. Tijd om hem te bouwen.

Voor bollebozen: procedurele effecten

Je kan nu zelf effecten maken. Deze bestaan aan het opleggen van kleur na kleur volgens je eigen samenstelling. Dit is leuk, en tof, maar wel veel werk! Er bestaat nog een andere manier om effecten te maken: procedures maken die een effect generen.

Procedurele effecten maken we door een functie te maken die bepaald welke kleur we tonen. We berekenen dus de kleur, waarna we ze tonen. Zo kun je een kleur maken die reageert op de buitenwereld, bevoorbeeld een temperatuursensor die aanleiding heeft tot een kleur tussen blauw en rood. Enkele ideeën voor procedurele effecten:

- Een willekeurige kleur tonen een aantal seconden, met dan een 'fade' overgang
- Laat de LED sneller en sneller blinken
- Laat de LED twee kleuren sneller en sneller blinken
- Geef de tijd sinds de Arduino aan is mee met een kleur
- enz ...

Laat ons de eerste idee uitwerken. Download de sketch n04_RGB_procedural, en bestudeer het. Vooreerst moeten we leren hoe we een willekeurig getal kunnen kiezen. Hiertoe voegen we in de setup() functie volgende lijn toe:

```
randomSeed(analogRead(0));
```

Een random getal is enkel volledige willekeurig indien we niet kunnen raden waar het start. Hiertoe moeten we de random getallen generator zaaien (seed in het Engels) met een begingetal. We doen dit hier door het lezen van de waarde op de niet gebruikte analoge pin 0: analogRead(0). Gezien deze analoge pin bij ons met niets geconnecteerd is, zal er enkel willekeurige storing erop zitten, welke we niet vooraf kunnen raden. Dit zal ons ongekend begin zijn, zodat we zeker zijn dat de serie helderheid die we zien telkens anders is als we de Arduino heropstarten. We kunnen nu een willekeurig getal bekomen met de functie:

```
random(256);
```

Deze Arduino functie genereert een willekeurig getal tussen 0 en 255.

We nemen een interval van 6 seconden. De eerste 3 seconden tonen we op de twee LED twee willekeurig gekozen kleuren. Bij de overgang naar het tweede deel van ons interval, kiezen we een derde kleur, en tonen we op LED 1 een fade overgang van kleur 1 naar kleur 2, en op LED 2 een fade overgang van kleur 2 naar kleur 3. Zie de loop() functie, en de nieuwe functie

```
void smooth_transitie(float fadeval);
```

Hier betekent float een reeel getal (dus een kommagetal). Het dient een waarde te zijn tussen 0 en 1. Bij 0 tonen we de eerste kleur, bij 1 de tweede, en een waarde ertussen toont een mengkleur, zodat we een mooi fade effect krijgen. De menging is (* is x in programmeertaal):

```
(1-fadeval) * oude_kleur + fadeval * nieuwe_kleur
```

De waarde van fadeval is hoever we zijn in het tweede deel van het interval, herschaald naar een waarde tussen 0 en 1:

```
(timer-interval/2) / float(interval/2)
```

Hier is float() een functie die een geheel getal wijzigt naar een kommagetal, dit moet hier om zeker te zijn dat het resultaat een kommagetal zou zijn, een deling van twee gehele getallen levert immers steeds een geheel getal op in Arduino, bv $10 / 4 = 2$ en $3 / 4 = 0$.

Verder worden de nodige variabelen gedefinieerd. Probeer het, en waag je aan je eigen procedure.

Fe LED Cube constructie

Schematisch

We maken nu de Fe LED kubus. Eerst leggen we de constructie uit aan de hand van een groot schakelbord. Indien je zo'n groot schakelbord hebt, kun je het onmiddellijk proberen. Op het schakelbord zijn de genummerde kolommen verticaal verbonden per 5 gaatjes. De bovenste en onderste 2 rijen zijn horizontaal verbonden.

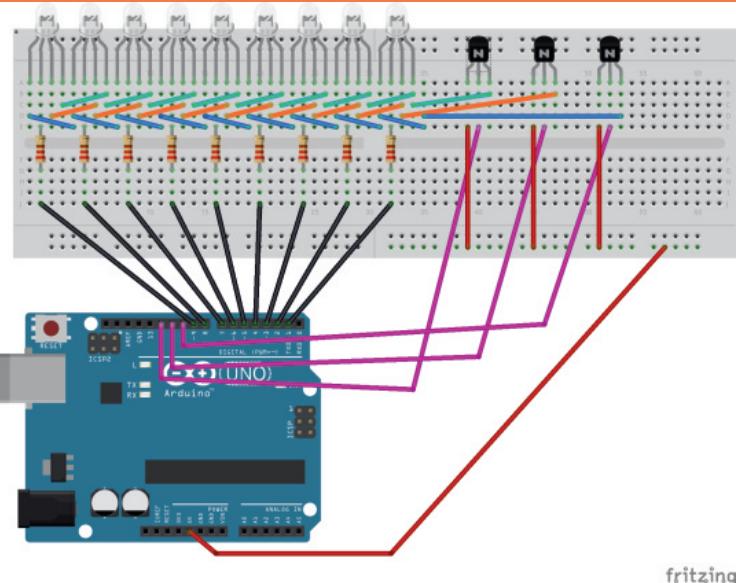
We zien dat de gelijke kleuren beentjes van de LEDs allemaal met elkaar verbonden moeten worden alsook met een NPN transistor. Deze transistor gebruiken we voor de kleurselectie, via uitgangen 10, 11 en 12 op de Arduino. De uitgangen 1 tot en met 9 gebruiken we om de gemeenschappelijke kathode (-) aan te sturen passend over een 220

Ohm weerstand. De NPN transistor rechterbeen gaat naar de 5V (+) output van de Arduino.

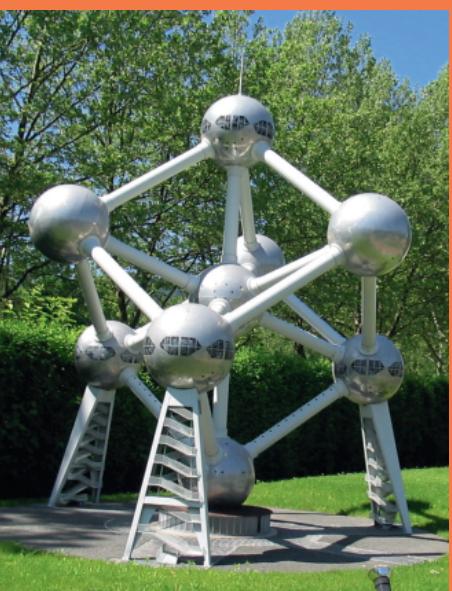
Je ziet dat er veel verbindingen zijn, maar dat het logisch opgebouwd is: 9 uitgangen voor 9 LEDs, en dan nog 3 uitgangen voor 3 kleuren. Zo zullen we $3 \times 9 = 27$ signaalcombinaties kunnen maken.

Indien je een groot schakelbord heb, kun je onmiddellijk de vorige Arduino sketch aanpassen om effecten te generen. Indien niet, geen probleem, we bouwen de kubus nu, en testen daarna!

Schakeling van de Fe Cube



Het Atomium



Fe Cube

We hebben 9 LEDs, en niet 8, het aantal hoekpunten van een kubus. Dit omdat we ze solderen in de vorm van een Fe Cubus. We noemen onze kubus een Fe Cubus omdat het georganiseerd is zoals het Ijzer (Fe) metaal kristalrooster. Ijzermetaal heeft een specifieke structuur welke uitgebeeld wordt in het Atomium gebouw in Brussel.

We noemen deze structuur een Lichaams-gecentreerd kubisch kristalrooster (in het Engels Cubic crystal system, genoteerd als bcc).

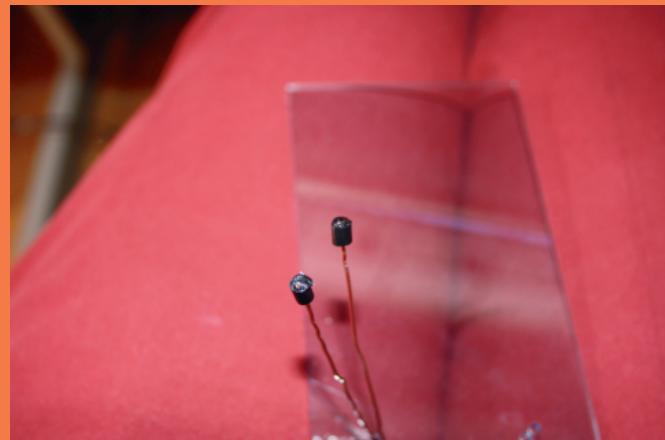
Constructie

1. Neem de lasercut delen, en zorg dat ze goed in elkaar passen. Het deksel bevat enkele gaten. Dit zijn de gaten voor de pilaren waarop de kubus komt. We zullen hem niet zoals het Atomium bouwen, maar als een gewone kubus. We hebben een pilaar nodig per LED, dus 4 hoeken met 2 gaten (voor 8 pilaren), en een centrumpilaar. Verifieer de afstanden, dit is de lengte van een zijde

Lasercut stukken



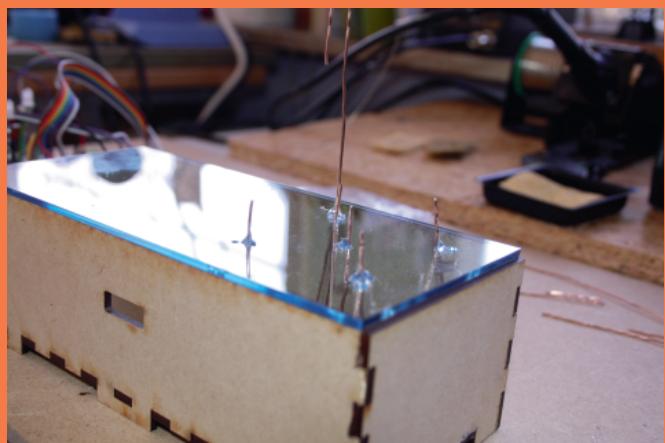
Pilaren



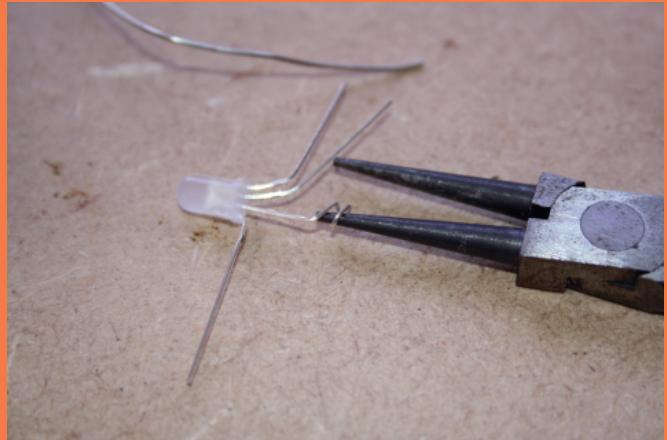
2. Maak de 9 stukken hard ijzer (normaal paperclips) recht, en maak er de 9 pilaren van door ze door de gaatjes te steken. van de kubus. Lijm aan de onderkant van het ijzer de plastieken stukjes vast. Zet het deksel op de doos, zodat de plastieken stukjes de grond van de doos raken. De plastieken stukjes zorgen ervoor dat de ijzeren staafjes niet te gemakkelijk rondschuiven. Lijm de pilaren vast aan het deksel.

Selecteer 4 hoekpilaren die je zal gebruiken voor de onderste 4 LED, en knip die af op enkele centimeter boven het deksel. Hierop zullen we de eerste 4 LED kathodes solderen.

Pilaren

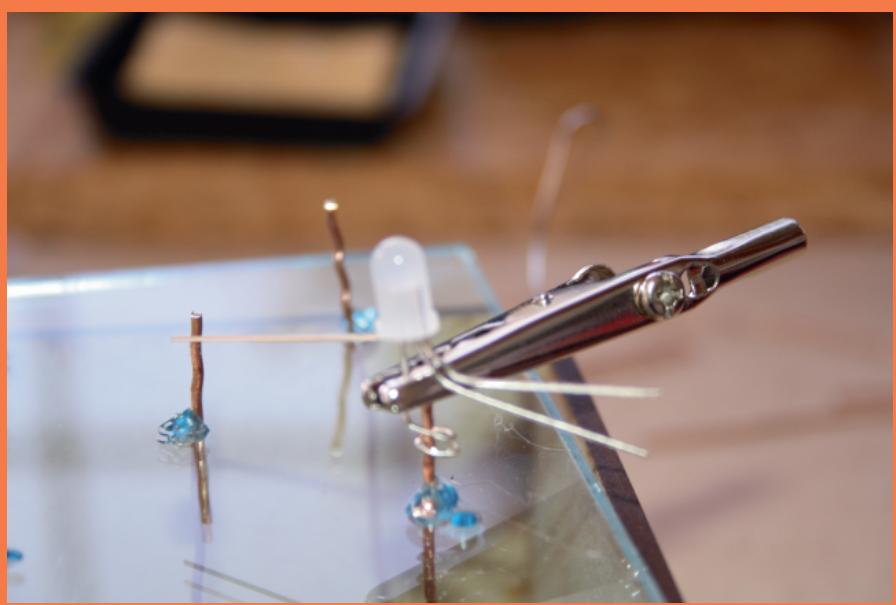


RGB LED



3. We bereiden nu de 9 RGB LEDs voor. Neem een LED, en plooït het been voor het rode licht horizontaal. Gebruik een afgeronde tang om een helix (krul) te maken in de gemeenschappelijke kathode (-). Herhaal dit voor de andere 8 RGB LED.

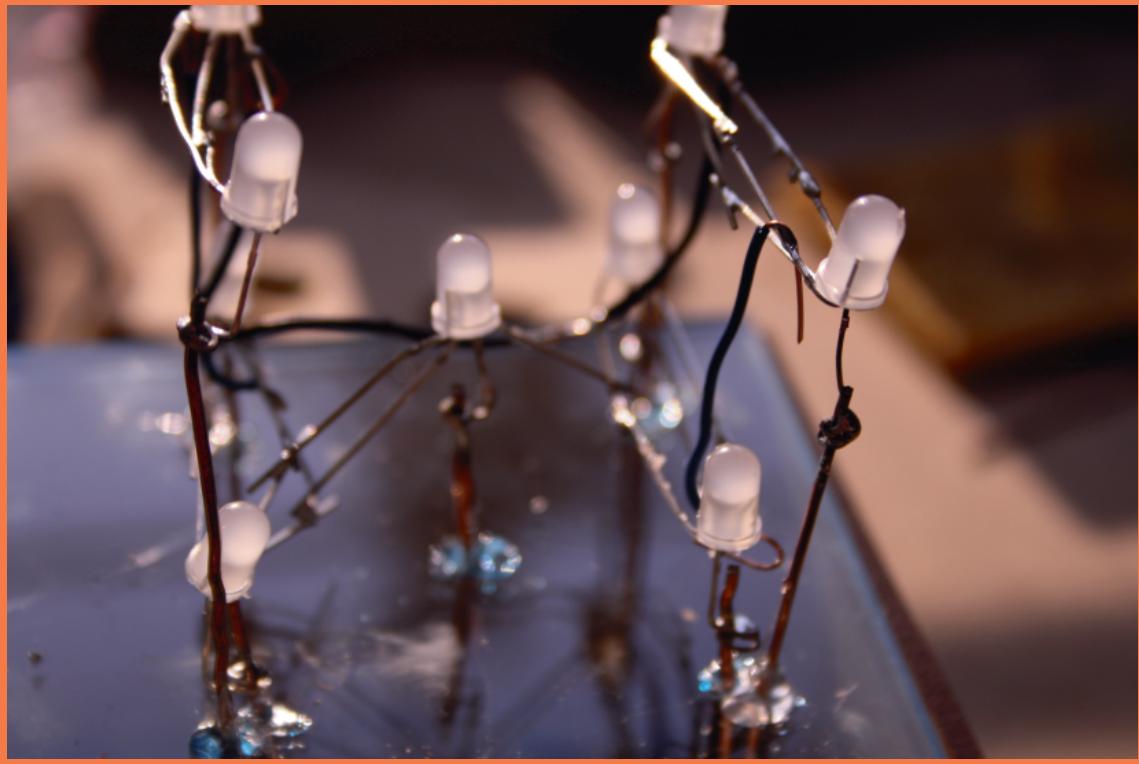
LED vastmaken



4. Schuif nu de krul over een pilaar, en hou het op zijn plaats met een klem. Soldeer de gemeenschappelijke kathode vast aan de pilaar. Opgelet, zorg dat de andere beentjes vrij blijven. Doe hetzelfde voor de ander 3 onderste LED. Knip de middelste pilaar op halve hoogte van de kubus, en soldeer de LED daar ook vast via de kathode. Doe daarna de bovenste 4 LED.

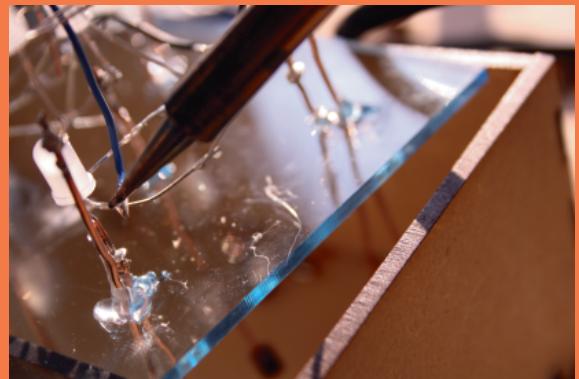
5. Je hebt nu al iets dat er als de kubus uitziet. We dienen nu alle gelijke LED kleurbeentjes met elkaar te verbinden. Er zijn verschillende manieren waarop je dit kunt doen. We stellen het volgende voor: Kies 2 LED op een zijde en verbind van die twee rood met rood, groen met groen, en blauw met blauw (gebruik eventueel een alcoholstift om te herinneren welk been welk kleur is). Gebruik daarna draadjes om de paarsgewijze LEDverbindingen met elkaar te verbinden.

De kubus krijgt vorm



6. Doe verder met draadjes toevoegen tot alle kleurenbeentjes onderling verbonden zijn. Om draadjes vast te solderen, knip je eerst een draadje op de gewenste lengte, en strip je de uiteinden. Vertin dan de uiteinden: op je werkblad leg je een laagje tin op het gestripte stuk. Verwarm dan met je soldeertip de plaats waar je het draadje aan vast wil maken. Hou je draadje op die plaats, en verwarm opnieuw. Voeg extra soldeertin toe met je soldeerbout indien nodig.

Verbindingsdraad



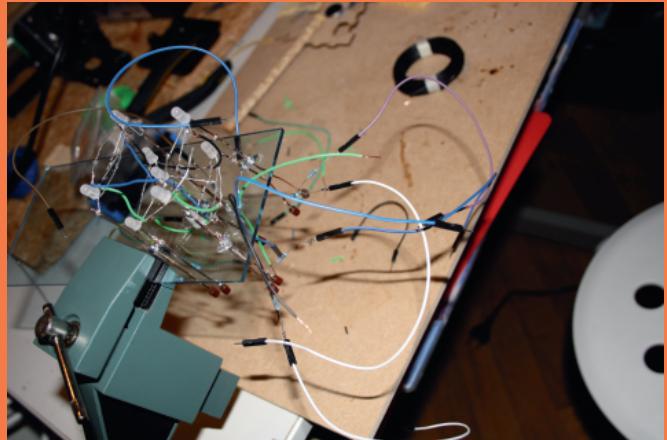
7. Zoals op het schema getoond werd, dient elke pilaar (de kathode) verbonden te worden aan de Arduino via een weerstand van 220 Ohm. Neem de weerstanden, en maak aan beide zijden een helix (krul) zoals je deed met het kathode been van de LED. De ene zijde van de weerstand soldeer je vast aan de draad die in de Arduino kan gestoken worden.

De andere kant van de weerstand soldeer je vast aan de onderkant van de pilaar, in de buurt van het plastieken stopstukje.

Als alles goed was, heb je nu 9 draden die in de Arduino uitgangen 1 tot en met 9 kunnen geplugged worden.

Doe dit als volgt. Kies een voorkant voor je kubus. Als je ernaar kijkt, zie je 3 richtingen. Van de bodem (B) naar de top

Weerstanden bevestigen aan de pilaren



(T), van links (L) naar rechts (R), en van voor (front, F) naar achter (A). Op die manier benoemen we de verschillende LED met codes als bevoorbeeld TLA: de top, links, achter LED. De LED in het centrum, noemen we evenwel MID, van midden. Verbind op volgende manier de LED met de verschillende Arduino uitgangen:

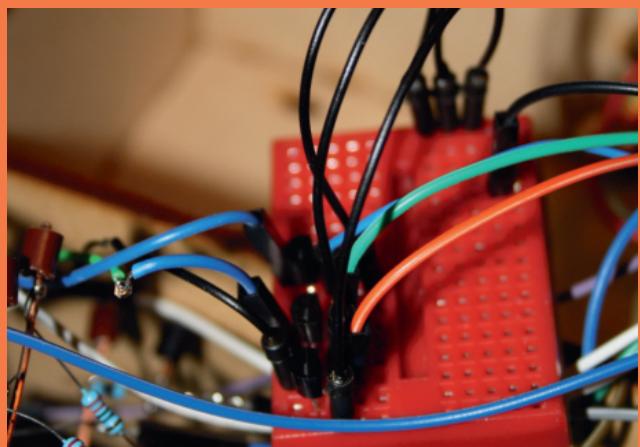
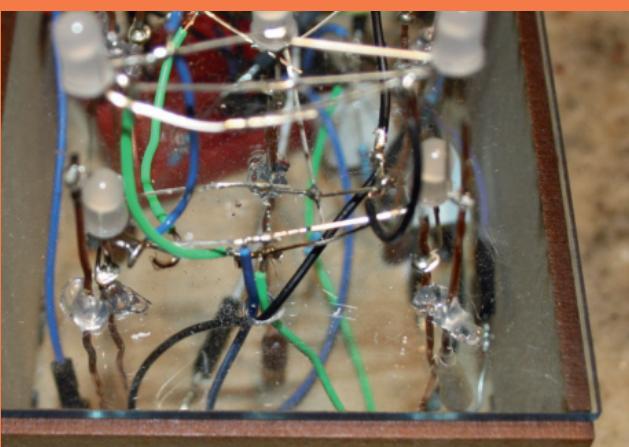
BLA: uitgang 1	TLA: uitgang 2
BLF: uitgang 3	TLF: uitgang 4
MID: uitgang 5	
TRA: uitgang 6	TRF: uitgang 7
BRA: uitgang 8	BRF: uitgang 9

8. Nu doen we de 3 kleur circuits. Neem een rode draad en verbindt hem met het circuit van de rode LED benen. Steek hem door het voorziene gat in het deksel. Doe hetzelfde met het circuit van de groene en blauwe LED benen.

Neem het kleine schakelbord, en plaats de NPN transistoren erop (idem aan vorige circuit). De collectoren verbindt je onderling naar een vrije lijn op het kleine schakelbord, welke je verbindt naar de 5V uitgang van de Arduino.

De uitgang van de eerste NPN transistor verbind je met het rode circuit, van de tweede met het groene circuit en de derde

De kleuren circuits naar het schakelbord met NPN transitoren



met het blauwe circuit.

Het middelste been van de eerste NPN transistor verbindt je met uitgang 10 van de Arduino. Het middelste been van de tweede NPN transistor met uitgang 11, en dat van de derde NPN transistor met uitgang 12.

9. De basis Fe Cube is hiermee af! Deze kubus is niet interactief, maar we kunnen reeds een programma schrijven om alle LEDs uit te testen. We doen dit eerst voor we verder bouwen.

Test Arduino Sketch

We nemen nu de vorige sketch die we gemaakt hebben, **n03_RGB_multiplex.ino**, en passen deze aan om onze Fe kubus te testen. Je kan de sketch downloaden van de website, het is sketch **n10_cube_test.ino**.

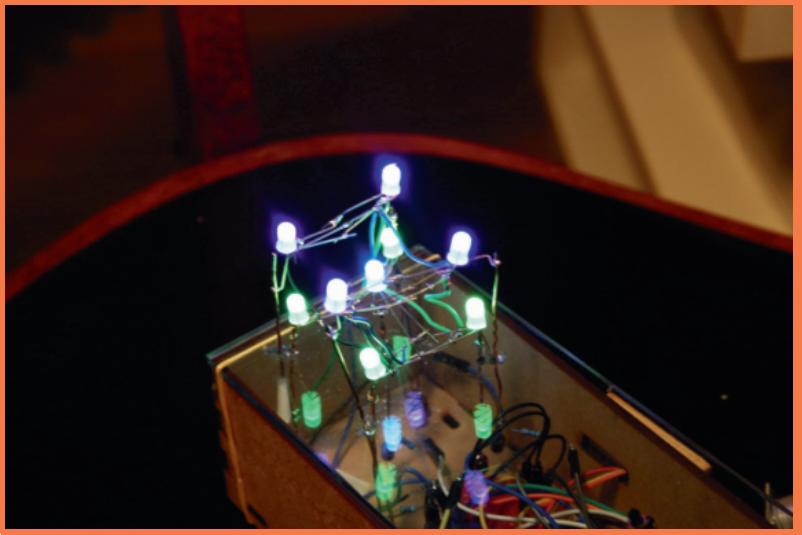
Je ziet hierin een beperkt aantal wijzigingen. In de setup zie je dat de verschillende kubus LED uitgangen gedefinieerd worden, bv

```
int ledBLA=1;  
welke we dan als een OUTPUT definieren, en op HIGH zetten.
```

Verder is de enige wijziging in de LEDs functie. We zullen alle bodem LEDs de eerste kleur geven, en de top LEDs de tweede kleur. De middelste LED geven we de witte kleur:

```
color(255, 255, 255);
```

De Fe Cube met eerste effect



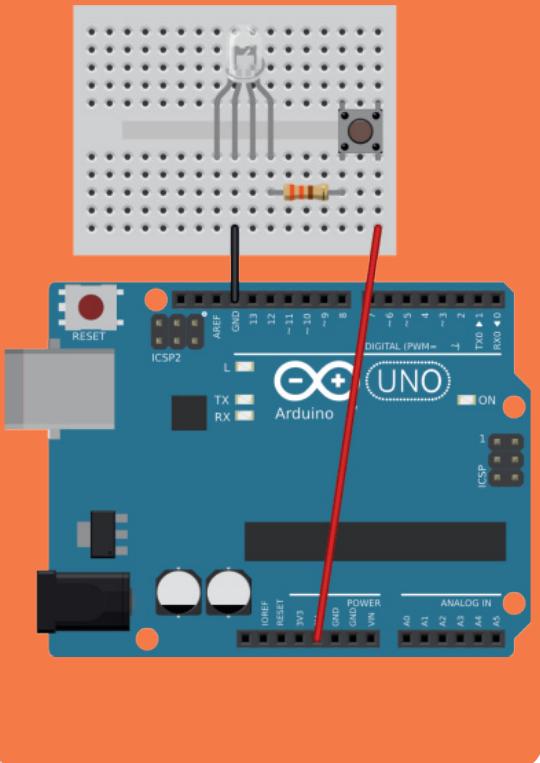
Opdracht 4

1. Wijzig de Arduino sketch zodat elke LED een eigen kleur heeft. Zie je de kleur waar je die verwacht?
2. Maak een eigen effect. Bepaal in de loop() functie in hoeveel delen je het interval zal opdelen. Maak dan in plaats van de LEDs() functie eigen functies LEDs1(), LEDs2(), ..., voor elke intervaldeel. In deze geef je de LED de kleur die je wil, om zo je eigen effect samen te stellen.

Drukknop voor interactie

Principe

Drukknop principe



Nu de basis kubus af is voegen we een drukknop toe. Dit laat ons toe om te interageren met de kubus via de knop. We beginnen met korte uitleg over de schakeling achter de drukknop.

Neem de drukknop. Je merkt 4 pinnen. Daarvan staan er twee dicht bijeen aan de ene kant, en twee dicht bijeen aan de andere kant. Zet de knop zo dat deze twee kanten links en rechts zijn. De pinnen die dicht bijeen staan zijn niet met elkaar verbonden als je de knop niet aanraakt. Druk je de knop in, dan worden deze twee pinnen met elkaar verbonden. De twee pinnen onderaan zijn steeds onderling verbonden, en ook de twee pinnen bovenaan zijn steeds verbonden.

Waarom 4 pinnen en niet 2? Dit is omdat het makkelijker solderen is zo. Je zal vaak meer dan 2 draden aan een knop moeten bevestigen, en dus zijn twee extra pinnen handig.

Net zoals onze allereerste schakeling tonen we hoe het werkt door de Arduino als batterij te gebruiken. Neem je kleine schakelbord, en maak het circuit zoals hiernaast afgebeeld. Test dat inderdaad bij indrukken de LED blauw oplicht, terwijl hij anders af staat.

Drukknop status lezen

We hebben nu geleerd hoe een drukknop de stroom kan onderbreken. Dit kan nuttig zijn in sommige omstandigheden, maar niet voor onze Fe Kubus. In de plaats willen we een knop waardoor we input kunnen krijgen van de gebruiker. In andere woorden, we willen weten als de gebruiker de knop drukt of niet, en we willen daarop reageren via Arduino code.

De Arduino heeft een speciale functie om input te lezen van een digitale pin:

`digitalRead(pin);`

Om deze methode te gebruiken moeten we een pin definieren als INPUT in de `setup()` functie:

`pinMode(pin, INPUT);`

De functie zal HIGH lezen als de pin op 5V staat en LOW als

ze op GND is. De truck is dus om een circuit te maken die LOW is als de knop niet ingedrukt is en HIGH als de knop ingedrukt is. Zo'n circuit is gegeven in weergegeven in de zijkalk. Maak het na. Let op, een van de weerstanden is $10\text{k}\Omega$, de andere 220Ω of 330Ω !

Laat ons dit circuit analyseren. De pin om de status van de knop te lezen is pin 12. Als de drukknop niet is ingedrukt zien we dat de pin verbonden is via zwart met de GND. Het zal dus in de GND staat of LOW zijn. Je ziet dat er een weerstand is, maar deze lijkt geen functie te hebben.

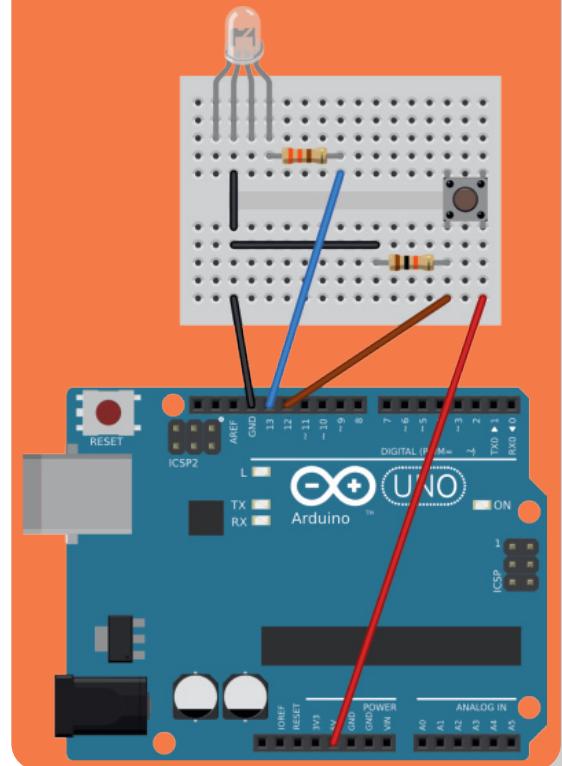
Beschouw nu het geval dat de knop ingedrukt is. Er is nu een connectie van links naar rechts in de drukknop. Zwart (GND) is dus verbonden met rood die naar de 5V op de Arduino gaat. Moest er geen weerstand zijn dan zouden we een kortsluiting krijgen: stroom die vrij stroomt van 5V naar GND. Aldus moeten we een weerstand hebben om kortsluiting te voorkomen. Welke weerstand hebben we nodig? Wel, we willen dat pin 12 op 5V staat als de knop ingedrukt is, en we willen niet veel stroom gebruiken om dit te verkrijgen. Bijgevolg willen we een grote weerstand!

Een grote weerstand zal zorgen dat de weerstand in de draden van de drukknop naar de 5V pin verwaarloosbaar is ten opzichte van de grote weerstand, en dus zal pin 12 op 5V staan. Terzelfdertijd zal een grote weerstand ervoor zorgen dat er maar heel weinig stroom zal vloeien over de drukknop, en ons circuit zal dus heel efficient zijn. Daarom nemen we als weerstand een kleine van $10000\ \Omega$, welke we ook schrijven als $10\text{k}\Omega$ of $10k\Omega$. We noemen zo'n weerstand op deze manier gebruikt een pull-down (naar GND) of pull-up (naar 5V) weerstand.

We kunnen nu Arduino code schrijven die de status van pin 12 leest, en het effect wijzigt op de LED na indrukken van de knop. Je vindt de code op de website als [n05_RGB_pushbtn.ino](#)

Deze sketch bevat enkele belangrijke wijzigingen. Eerst en vooral gebruiken we `#define` om nummers aan pinnen toe te kennen. Merk op dat op het einde van deze lijn niet langer ; staat. `#define` is niet beter, gewoon een alternatief die wel minder geheugen in beslag neemt. Je kan het gebruiken voor alle constante waarden.

Drukknop status lezen



Daarnaast maken we gebruik van lijsten. We geven op hoeveel effecten we zullen maken:

```
const int nreffects = 4;
```

Dan maken we een lijst die al onze effecten opsomt. De syntax is moeilijk, maar je dient enkel een eigen effect hieraan toe te voegen, of weg te nemen:

```
void (*effects[nreffects])() = {effect1, effect2,  
effect3, effect4};
```

Voor elk effect houden we dan nog een lijst bij van hoe lang het effect duurt:

```
long intervals[nreffects] = {3000, 3000, 3000, 3};
```

Zoals eerder herhaalt een effect zich eindeloos lang. Met de drukknop zullen we ervoor zorgen dat na drukken het volgende effect getoond wordt. Je ziet dit aan het begin van de loop() functie: We lezen de waarde op de drukknop pin, en als deze ingedrukt is (HIGH), en eerder was hij niet ingedrukt (buttonStatePrev == LOW), dan verhogen we het effect dat getoond wordt met 1:

```
showeffect += 1;
```

We zorgen er natuurlijk wel voor nooit meer effecten te hebben dan we gemaakt hebben. Lijsten beginnen te tellen bij 0. Onze lijst bevat 4 effecten, dus: 0, 1, 2 en 3. Daarom, als we effect met nummer 4 willen tonen, zorgen we dat we opnieuw bij 0 beginnen:

```
if (showeffect >= nreffects) {  
    showeffect = 0;  
}
```

Daarna tonen we de effecten. Deze code is analoog aan wat je eerder deed. Enige verschil is dat we nu vooraf 4 effecten gemaakt hebben: blinken, licht aan, licht af, en licht gedimd.

Events

Wat kunnen we doen met maar een knop? Meer dan je denkt! Denk aan een aanraakscherm op een tablet. Wat kun je doen met een vinger? Denk aan de computermuis. Wat kun je doen met een knop?

Ja, het antwoord is: veel! Je kan een normale druk hebben, een lange druk, een dubbele druk Je zou zelf kunnen reageren op basis van bepaalde patronen (bv SOS in morse). Het verschillend reageren op verschillende drukken is een kwestie van software. In onze sketch is het een kwestie van bij te houden op welk tijdstip de knop ingedrukt werd, en wanneer hij losgelaten wordt. Het programma die we zullen gebruiken voor de kubus zal in staat zijn om onderscheid te maken

tussen een aantal type drukken.

Opdracht 5

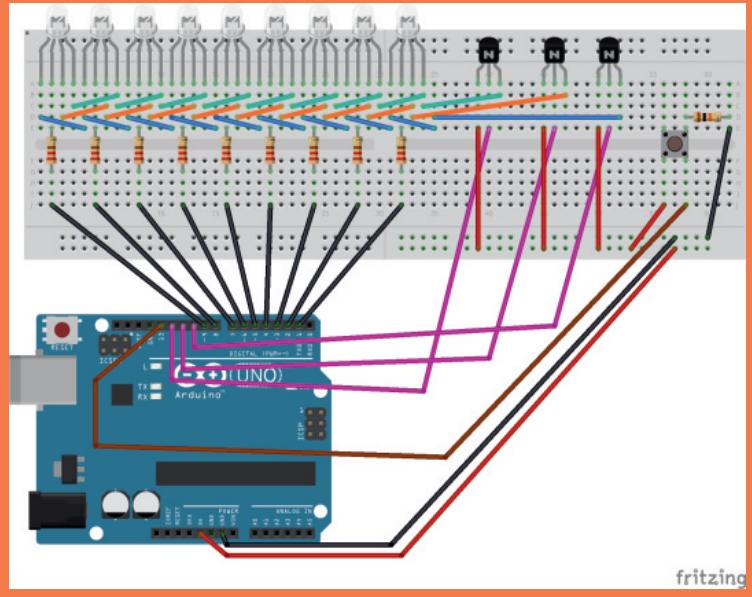
1. Pas vorige Arduino sketch aan: voeg een eigen effect toe. Jouw manier van flikkeren. Zie je het effect die je verwacht? Voor deze opdracht dien je een functie effect5() te schrijven, en deze toe te voegen aan de lijsten.
2. Maak een effect dat de eerste letter van je naam in Morse code toont. Maak een tweede effect voor de tweede letter. Doe verder tot je alle letters van je naam hebt. Gebruik de drukknop om tussen de letters te schakelen. Zoek hiervoor op internet naar de Morse code van de letters die je nodig hebt.

De Fe Cube afwerken

Drukknop inbouwen

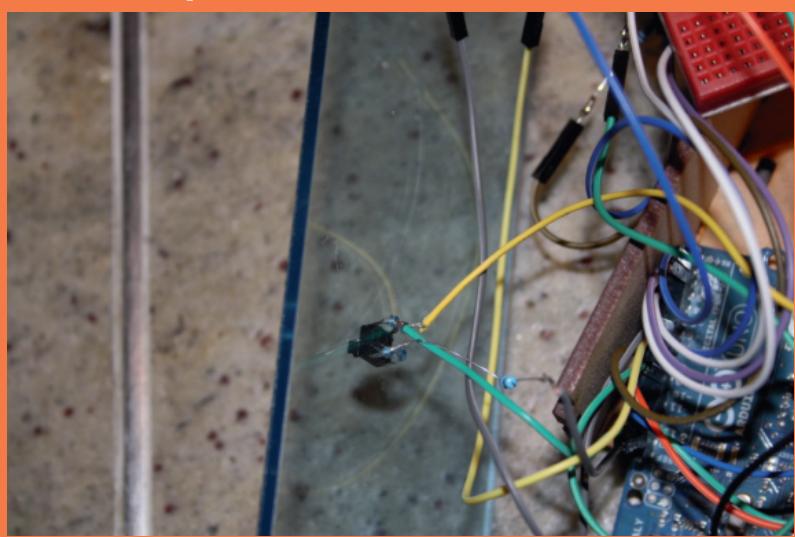
Na vorige sectie heb je op je schakelbord de drukknop staan. Je dient deze schakeling over te zetten naar het deksel van de

Fe Cube + drukknop schematisch



die je verbindt met de 5V uitgang van de Arduino. Gebruik die lijn op het schakelbord ook voor de draad van de drukknop die op 5V moet staan.

Drukknop vastmaken



Je hebt nu alle elementen, en deze zijn allemaal juist verbonden. Sluit het geheel, schroef de Arduino vast op de bodem met de voorziene schroeven zo dat je eenvoudig de USB kabel kan vast- of losmaken. Maak het geheel tijdelijk vast met elastieken. Na uittesten of alles werkt kun je de doos toelijmen.

Fe Kubus uittesten

Onze kubus test (`n10_cube_test.ino`) zullen we nu aanpassen zodat we via de drukknop een effect kunnen kiezen. We maken volgende effecten:

1. Het effect uit n10_cube_test.ino
 2. Een slang effect waarbij alle LEDs een per een opgelicht

worden.

Je vindt de code op de website als [n11_cube_pushbtn_test.ino](#). Download deze code, en laad ze op je Arduino. Test of de drukknop je doet veranderen van effect. Test ook of de beweging van de slang overeenkomt met wat in de sketch staat.

Als je de sketch bekijkt in de Arduino IDE zie je dat het een combinatie is van de vorige kubus test code, en de drukknop code. Het effect van de kubus test code is nu aanwezig in de functie effect1().

Het slang effect staat in de functie snake(). Je zou deze functie nu in staat moeten zijn volledig te verstaan. De loop() functie is analoog aan deze uit de drukknop sketch. In de lijsten effects en intervals, zie je dat we twee effecten hebben (nreffects=2;), en dat de eerste 2000 milliseconden duurt, en de tweede 9000.

Alles werkt? Proficiat. Je Fe Kubus is af. Zorg dat alles er verzorgt uitziet, en lijm het deksel vast.

De afgewerkte Fe Cube



Opdracht 6

1. Je had eerder je eigen effect gemaakt op de kubus. Voeg dit toe als een effect. Of vervang effect1() door dit eigen effect.
2. Wijzig de kleur van de slang in het snake() effect.
3. Maak je hoogstpersoonlijke effect, en voeg het toe aan de lijst. Heb je een mooi effect? Stuur de functie van het effect naar ons, mchristina.ciocci@ingegno.be, en we posten het!

De Fe Cube Bibliotheek

Waarom?

Je bent erin geslaagd. De Fe kubus is af, en je kan effecten tonen. Opdracht volbracht. Als je het niet erg vindt, doen we nog even verder evenwel.

We kunnen nog veel meer met onze kubus doen, maar je zal wel opgemerkt hebben dat effecten schrijven veel werk is. Je moet de LED aanzetten, dan weer af. Intervallen uitrekenen. Dat wordt vlug veel tikwerk.

We kunnen verschillende functies schrijven die een groot stuk van het werk overnemen, zodat je kunt concentreren op zo mooi mogelijke effecten schrijven. Deze functies hebben we al voor jou gemaakt, en gebundeld in wat men een 'Bibliotheek' (Engels: Library) noemt. Je downloadt deze bibliotheek en voegt ze toe bij je effect, en kan van alle functies gebruik maken.

Principe van de Library

De library werkt anders dan de code die we tot nu toe geschreven hebben. Hij is gebaseerd op het principe van frames. Frames ken je misschien uit de filmwereld. Het is een enkel beeld, welke 1/25 van een seconde getoond wordt. Via persistentie van visie krijg je dan een bewegend beeld.

De library zal je effect oproepen en vragen een specifiek frame terug te geven. Het voordeel voor onze code is dat je effect maar een keer moet uitrekenen wat in een frame moet komen. Dat is efficienter dat wat we tot nu deden: telkens opnieuw alles uitrekenen.

Je deelt dus je effect op in frames, en geeft de frames door aan de functie die de effecten toont. Wat dien je dan door te geven? Wel, we hebben 9 LED, met 3 kleuren, dus dienen we een lijst van 27 getallen door te geven.

Eenvoudig voorbeeld

Laat ons de vorige sketch herschrijven in termen van de bibliotheek. Je vindt de code op de website als **n12_cube_library_test.ino**. In de map van deze ino file (map **n12_cube_library_test**) dien je ook de bibliotheek **fecube.h** op te slaan, welke je ook op de website vindt. Als je nu de Arduino IDE opent voor deze sketch zul je twee tabbladen

zien, een voor de ino file, en een voor de bibliotheek,

Het bestand **fecube.h** bevat de bibliotheek. Je dient dit file niet te begrijpen, je moet enkel weten welke functies erin staan die je kan gebruiken om de Fe Kubus aan te sturen. We noemen dit een 'zwarte doos' aanpak: een bibliotheek biedt diensten aan die we gebruiken maar waarvan we de interne details niet kennen of zien (daarom, zwarte doos).

Bekijk de file **n12_cube_library_test.ino**. Je ziet bovenaan de lijn:

```
#include "fecube.h"
```

Dit laadt de bibliotheek uit het bestand fecube.h die in dezelfde bestandsmap als onze sketch zit. We kunnen nu alle functies uit de bibliotheek gebruiken.

Daarna schrijven we de twee effecten op de manier die de bibliotheek ze wil. Alle effecten dienen dezelfde parameters te hebben: framenummer en frame[27]. Bv voor effect1():

```
void effect1(unsigned long framenum, int frame[27])
```

Je effect dient het frame dat hoort bij een bepaald framenummer te berekenen, en in de variabele frame te stoppen. Hiertoe moet je weten in welke orde je je LED kleuren moet stoppen in de frame. De orde van de LED in de frame is:

```
{ledTLF, ledTLA, ledTRF, ledTRA, ledBLF,  
ledBLA, ledBRF, ledBRA, ledMID}
```

Bij effect1() zie je dat we een frame maken die de kleuren bevat die we willen, en met de functie fecube_set_frame() laden we dit frame in het gewenste frame.

In plaats van te rekenen met interval rekenen we nu met frame nummers. We hebben 25 frames per seconde, dus als we de eerste frame willen tonen voor 1 seconde, dienen we ervoor te zorgen dat de code 25 frames zichtbaar is:

```
if (framenum <=25) {
```

In het andere geval tonen we het tweede frame.

Ook het effect snake() is op deze manier herschreven. Om geen lange frames te moeten tikken zoals in effect1() gebruiken we de bibliotheekfunctie fecube_clearframe om de frame waarden allemaal op 0 te zetten (geen kleur), en dan gebruiken we fecube_set_ledcolor functie om een bepaalde LED een opgegeven kleur te geven. Je gebruikt hier de bibliotheekwaarden cBLA, cBLF, ... om de LED aan te duiden die je een kleur wil geven.

We hebben hiermee de twee effecten herschreven. Vervolgens maken we een lijst met de effecten, alsook met de

duur van elk effect:

```
const int nreffects = 2;
int showeffect = 0;
void (*effects[nreffects])(unsigned long, int[27])
= {effect1, snake};
unsigned long effect_duration[nreffects] = {2000UL, 9000UL};
```

We dienen nu enkel nog de bibliotheek te laten weten welke effecten we willen. We doen dit in de setup code, welke de bibliotheekfunctie **fecube_setup()** gebruikt als de kubus te initialiseren, en **fecube_set_effects()** om door te geven welke effecten te tonen. Onze loop() functie dient enkel de bibliotheekfunctie **fecube_loop()** op te roepen.

Laadt deze sketch op naar je Fe Kubus, en stel vast dat alles nog werkt zoals eerder. Evenwel heb je wel enkele extra functionaliteiten: druk je de knop twee keer snel na elkaar in, of druk je heel lang in (meer dan 5 sec) dan schakelt de kubus af. Bij dubbelklik duurt het afzetten 10 seconden, anders blijft de kubus af tot je opnieuw lang drukt. Probeer het!

Opdracht 7

Herschrijf je eigen effect uit opdracht 6 in termen van de fecube bibliotheek. Heb je een mooi effect? Stuur de functie van het effect naar ons, mchristina.ciocci@ingegno.be, en we posten het!

Volledig voorbeeld

We eindigen met een voorbeeld die alle mogelijkheden van de fecube bibliotheek toont. Download de sketch **n13_cube_showtime.ino** van de website.

Dit voorbeeld bevat verschillende effecten die je ook moet downloaden en in dezelfde map (**n13_cube_showtime**) plaatsen:

```
#include "effect_fixed_random_smooth.h"
#include "effect_rolldice.h"
#include "effect_beating_heart.h"
#include "effect_siren.h"
#include "effect_snake.h"
#include "effect_simple_pattern.h"
```

Al deze files dien je dus ook te downloaden.
Verder heb je natuurlijk ook de bibliotheek `fecube.h` nodig.

De sketch bevat 9 effecten. Je ziet ze in de lijst effects opgesomd. In de setup code hebben we nog enkele initialisatie functies:

```
set_fixed_color(255, 0, 255);
// timings for smooth color transitions
set_smooth_color_durations(5000, 10000);
```

Dit zijn functies waarmee je bepaalde effecten kunt aanpassen. Zo kun je met `set_fixed_color` bepalen welke kleur het effect: `effect_fixed_color()` zal tonen.

De code is redelijk simpel. Bollebozen kunnen de effecten van nabij bestuderen om ze te doorgronden. Dat is al wat moeilijker. De bibliotheek zelf doorgronden dien je maar te kunnen als je volwassen bent en programmeur wil worden! Wij eindigen met het effect **effect_simple_pattern()**, welke je in staat zou moeten zijn aan te passen. Open dus het bestand `effect_simple_pattern.h`, en lees de code. Het belangrijkste deel is het patroon:

Dit patroon bestaat uit 4 lijnen. De laatste lijn is de afsluiter van het patroon, en dient altijd aanwezig te zijn. De drie lijnen daarboven zijn 27 nummers die de kleur aanduiden in groepen van 3 per LED, en een 28e nummer die duur is dat dit patroon moet getoond worden. In dit geval, alle drie 1000 milliseconden, dus 1 seconde.

Onder het patroon staan enkele variabelen die je kan wijzigen,

Opdracht 8

Pas PatternSimple aan: wijzig een lijn door andere kleuren te tonen. Voeg enkele lijnen toe om je eigen effect te maken.

maar dit hoeft niet. Deze variabelen wijzigen hoe het patroon zich gedraagt. Daaronder staat dan een laad functie die je niet dient te wijzigen, en het effectieve effect, de functie **effect_simple_pattern()**, welke je ook niet dient te wijzigen.

Andere patronen maak je dus door enkel en alleen de **PatternSimple** lijst aan te passen.

In de sketch **n13_cube_showtime.ino** laden we ook een speciaal effect bij dubbelklik op de drukknop:

```
fecube_set dblpress_effect(rolldice_duration,  
                           effect_rolldice);
```

Hiermee geven we aan dat bij dubbelklik we het effect **effect_rolldice()** willen starten. Dat is een leuk effect dat voor jou 3 gewone dobbelstenen simuleert. Wil je een ander effect bij dubbelklik? Wijzig dan gewoon deze lijn in je sketch om een ander effect te laden.

Opdracht 9

Maak je eigen versie van n13_cube_showtime. Beslis hoeveel effecten je wil tonen, en welk effect je wil zien bij dubbelklik.

Om je eigen effecten te maken is kun je een effect maken zoals je deed in sketch n12_cube_library_test.ino, of je kan effect_simple_pattern.h aanpassen. Wil je verschillende eigen patronen? Sla dan effect_simple_pattern.h op met een nieuwe naam, bv effect_eigen_pattern.h. Wijzig bovenaan in twee lijnen EFFECT_SIMPLE_PATTERN in een eigen nieuwe unieke naam, bv EFFECT_EIGEN_PATTERN. Geef ook de twee functies load_simple_function en effect_simple_pattern een eigen unieke naam. In de functie effect_simple_pattern dien je dan ook ervoor te zorgen dat je load_simple_function wijzigt in de nieuwe naam die je die functie gegeven hebt.

Geen inspiratie voor nieuwe effecten, kijk dan eens op onze website welke effecten er allemaal zijn: <https://github.com/ingegno/fecube/tree/master/effects>

Heb je een eigen super effect? Stuur het ons, en we nemen het op in de lijst van effecten.

Ontdek ook onze andere producten op de ingegno.be shop! Maak je eigen Kurk-Engel of Kurk-Duivel. Maak een windmolen. En meer!



Gemaakt te Drongen, 2014
Door Ingegno.be