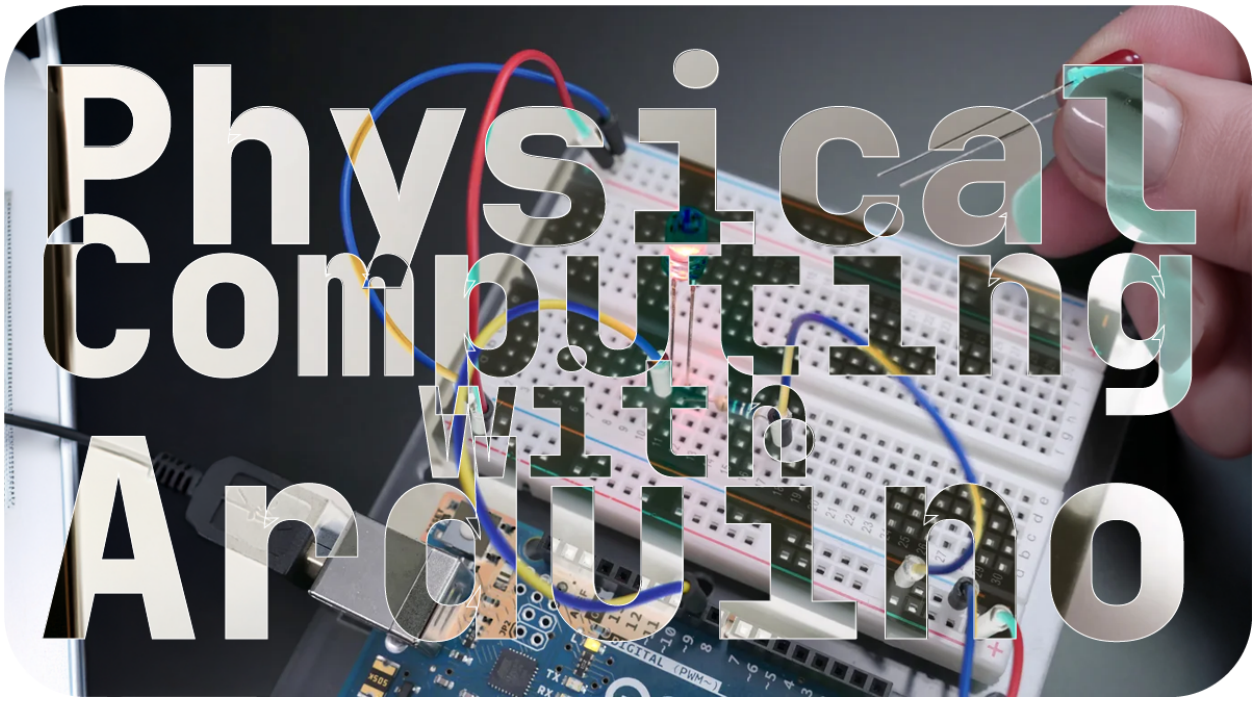



1. **Arduino: de omgeving en het platform**
    1. 1. Het Arduino platform
    1. 2. Het Arduino board
    1. 3. De software (IDE)
2. **Output: Arduino in actie**
    2. 1. Hello World!
    2. 2. Lingua Arduino
    2. 3. Speel met de parameters
    2. 4. De Autocomplete-feature van IDE 2.x
    2. 5. Andere actuators
3. **Circuits bouwen: een LED aansturen**
    3. 1. Wiring diagrams & schematics
    3. 2. Componenten
4. **Inputs: Arduino laten reageren**
    4. 1. Druk op de knop
    4. 2. Een on/off-button die vast blijft staan
    4. 3. Andere on/off-sensoren
5. **Variabele output: PWM en servo motoren**
    5. 1. Een LED dimmen met PWM
    5. 2. Input naar output
    5. 3. Servo motor control
6. **Additional Learning resources**

In deze tutorial leer je de basis van **physical computing met Arduino**. Je maakt kennis met de basisprincipes van elektronica, leert circuits lezen en opbouwen, en leert hoe je Arduino programmeert.

**Physical computing** is het ontwerpen en maken van interactieve systemen waarbij hardware, software en de fysieke wereld met elkaar in wisselwerking staan. Via sensoren kan een systeem informatie uit zijn omgeving waarnemen, terwijl actuatoren daarop reageren door bijvoorbeeld licht, beweging of geluid te genereren.

Arduino is een open source-platform voor physical computing, gebaseerd op een eenvoudig input/output (I/O) board en een computerprogramma.

# 1. Arduino: de omgeving en het platform

## 1.1. Het Arduino platform

Arduino bestaat uit twee belangrijke onderdelen:

1. **het Arduino board**: de hardware waarmee je aan de slag gaat wanneer je objecten bouwt.
2. **de Arduino-software**: een editor (IDE) die op je computer draait of in een webbrowser. In deze editor schrijf je een *sketch* (een computerprogramma met instructies) die je naar het Arduino board uploadt. Dit programma vertelt het board wat het moet doen.

De Arduino-website is een goede plek om projecten met Arduino te ontdekken, te leren en hulp te vragen.

- Leer de basis van Arduino via de tutorials in de verzameling Built-in Examples.
- Een uitgebreide verzameling tutorials.
- Op Project Hub vind je projecten van de Arduino-community.
- Het Forum.

## 1.2. Het Arduino board

Het Arduino board is een kleine elektronische schakeling met een microcontroller: een computerchip die je kunt programmeren om bepaalde taken uit te voeren. Die is veel minder krachtig dan je laptop, maar ook veel goedkoper en bijzonder handig om praktische toepassingen te bouwen.

Er zijn heel wat verschillende Arduino boards op de markt. Alle officiële boards vind je hier. In het atelier werken we vooral met de Arduino UNO Rev3, een van de meest gebruikte Arduino boards.

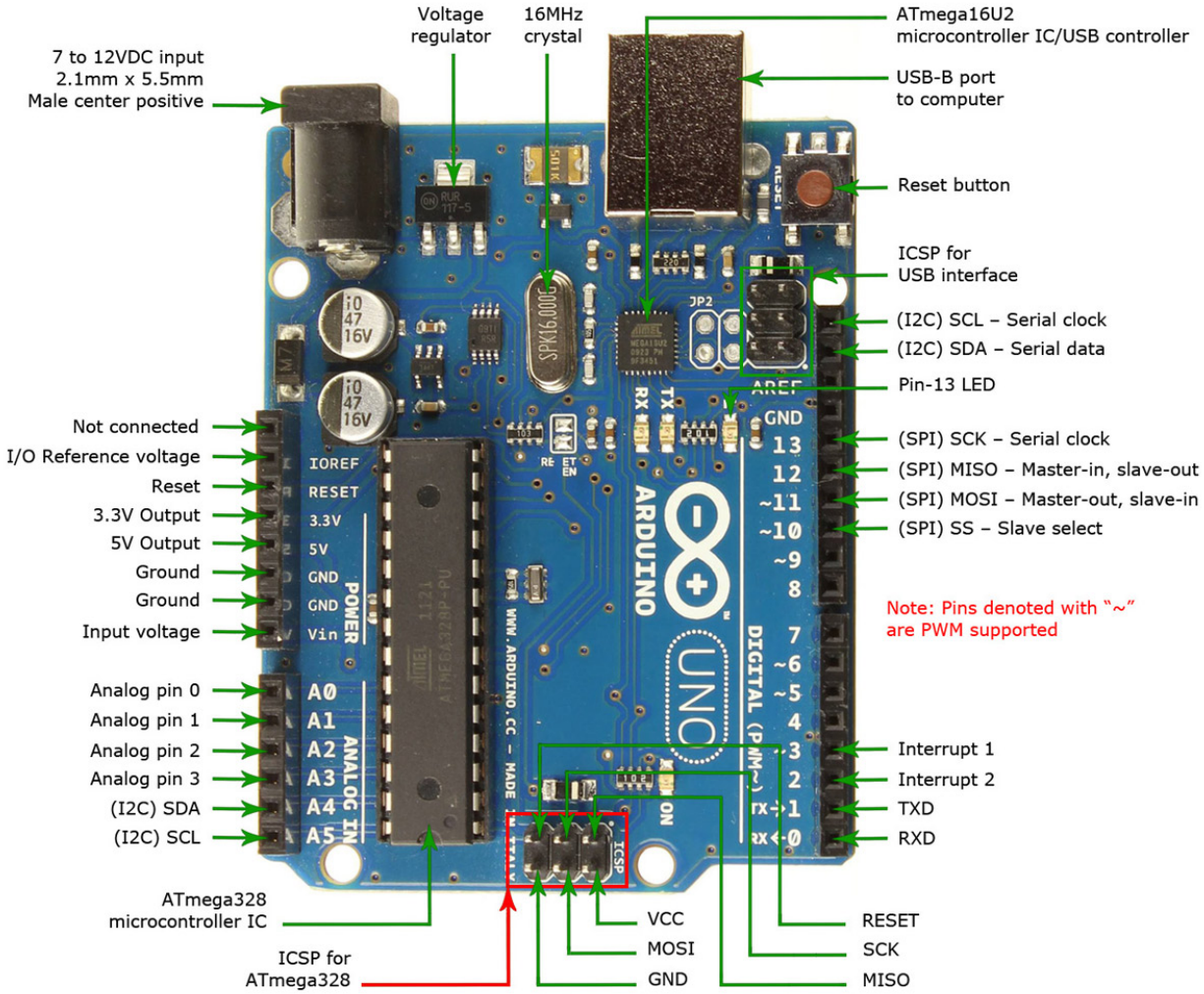



Op het Arduino board zie je een zwarte chip met 28 *pootjes*. Dat is de ATmega328P, het hart van je board.

De Arduino UNO Rev3, met de ATmega328P-microcontroller, beschikt over:

- **14 digital I/O pins** (pins 0–13). Deze kunnen als input of output worden gebruikt, afhankelijk van wat je in je sketch instelt.
- **6 analog input pins** (pins A0–A5). Deze pins ontvangen analoge spanningen (0–5V), bijvoorbeeld van een sensor. De ADC (Analog-to-Digital Converter) zet die spanning om in een getal tussen 0 en 1023, oftewel 1024 mogelijke waarden (10 bit).
- **6 PWM pins** (pins 3, 5. 6, 9, 10 en 11). Dit zijn digital pins waarop je PWM-signalen kunt genereren. Ze zijn aangeduid met een `~` (tilde).

- Je kunt het board **voeden** via de USB-poort van je computer (5V), via de meeste USB-opladers of via een externe voedingsadapter (7–12V aanbevolen, met een 2,1 mm barrel plug en positieve center pin).
- Daarnaast bevat de Arduino UNO Rev3 een ATmega16U2, die fungeert als **USB-to-serial converter**.
- ...




De Arduino-referentie vind je hier.
Het volledige schema van de Arduino UNO vind je hier.

## 1.3. De software (IDE)

De programma's die je voor Arduino schrijft, worden *sketches* genoemd. Je schrijft ze in **C/C++** met behulp van een code editor. Het Arduino-team biedt drie manieren om je board te programmeren:

- Een **IDE** (*integrated development environment*): software die je op je computer installeert en waarmee je code schrijft, compileert en uploadt.
- Een **web editor**, waarmee je online code schrijft. Je hebt een account nodig om je sketches online op te slaan.
- De **CLI** (*Command Line Interface*), voor wie liever via de command line werkt dan met een grafische interface.

We downloaden en installeren eerst de Arduino IDE 2.x. Daarna bekijken we hoe je een sketch vanuit de IDE naar het board uploadt.

## Installatie van de Arduino IDE

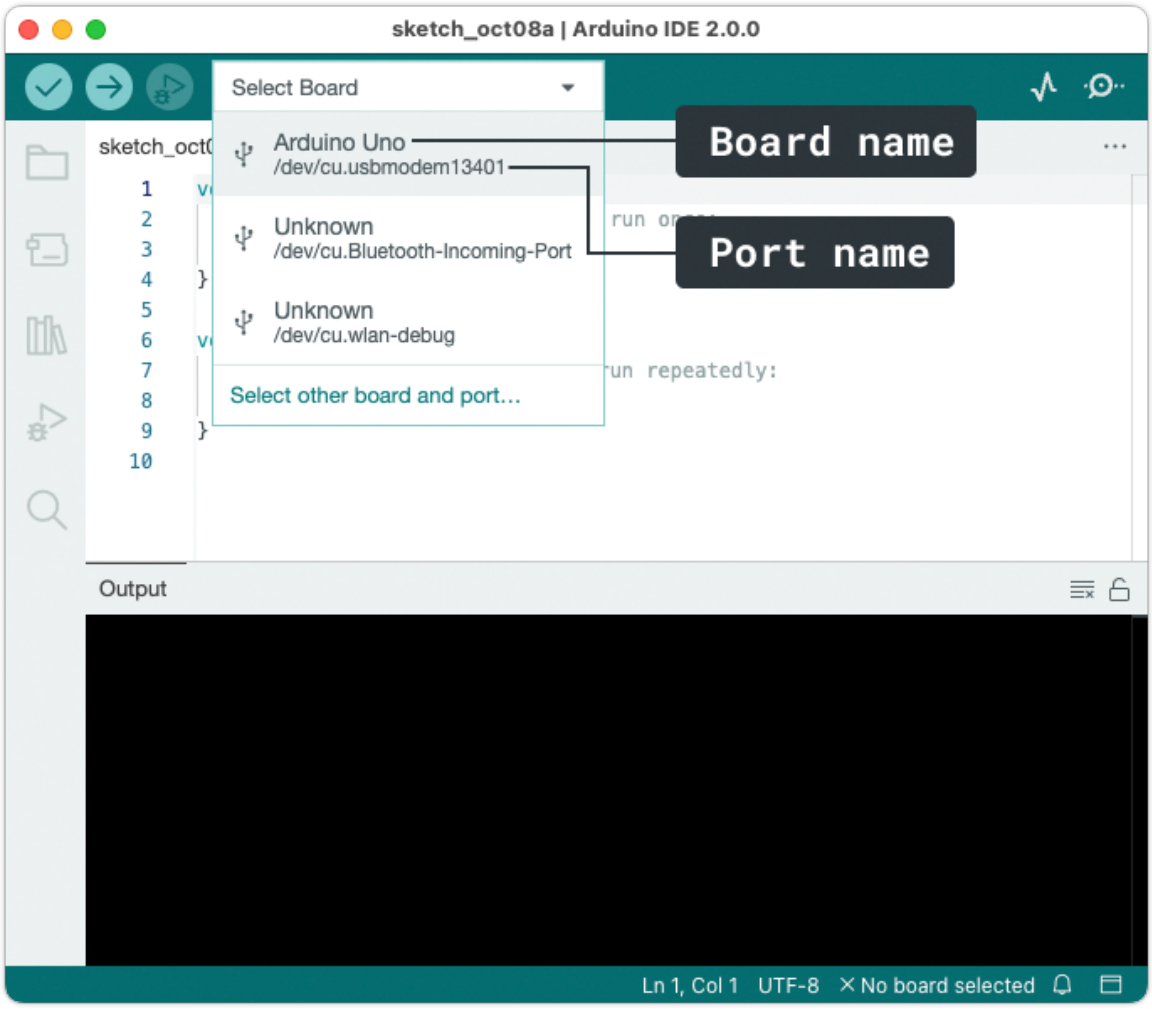
Download de meest recente stabiele versie voor jouw besturingssysteem via de Arduino-software Page. Je kunt ook je board selecteren op de Arduino-website en vervolgens de bijbehorende **quickstart**-gids volgen.

## Een sketch uploaden met de Arduino IDE 2.x

De manier waarop je de software instelt en met je Arduino board verbindt, verschilt enigszins afhankelijk van je computer en het board dat je gebruikt.

Voor je iets uploadt, moet je eerst **het board selecteren** waarmee je werkt.

Naast de knoppen **Verify** en **Upload** vind je een drop-down menu waarin je doorgaans de aangesloten Arduino boards kunt selecteren. Wordt je board niet automatisch herkend, klik dan op **Select other board and port...** en volg de instructies. Je weet dat het board verbonden is wanneer de naam ervan **vetgedrukt** verschijnt.




Om **code te uploaden**, kijken we eerst naar de toolbar bovenaan de editor.

Helemaal links staan twee knoppen: een **vinkje**, waarmee je de code *verifieert*, en een **pijl naar rechts**, waarmee je de code *uploadt*.

**Verify** controleert je sketch op fouten en compileert de code. **Upload** doet hetzelfde, maar stuurt de gecompileerde code vervolgens ook naar het board.




**Tip**

Als afsluiting van dit hoofdstuk kun je deze gids volgen. Je vindt er de basis van de Arduino IDE 2.x, een uitgebreid overzicht van de UI en links naar bijzondere functies zoals autocompletion en debugging.

# 2. Output: Arduino in actie

## 2.1. Hello World!

De Arduino-versie van *Hello World!* is een knipperende LED.

Je hebt alleen een Arduino board en een USB-kabel nodig.

Open een nieuwe sketch in de IDE. De onderstaande code staat er al in. Ze vormt de basis van elk Arduino-programma. Daar komen we later op terug.

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Geef het bestand een naam en sla het op.

Typ vervolgens de onderstaande code in de editor. Je mag de regels die beginnen met `//` overslaan: dat zijn commentaarregels.

```
// LED connected to digital pin 13 (UNO)
const int ledPin = 13;

// the setup function runs once when you press reset
// or power the board
void setup() {
  // initialize digital pin 13 as an output
  pinMode(ledPin, OUTPUT);

}

// the loop function runs over and over
void loop() {
  // turn the LED on (HIGH is the voltage level)
  digitalWrite(ledPin, HIGH);
  // wait for 1000 milliseconds or 1 second
  delay(1000);
  // turn the LED off by making the voltage LOW
  digitalWrite(ledPin, LOW);
  // wait for another second
  delay(1000);
}
```

Klik op de **Verify**-knop om te controleren of je code correct is.

Als alles goed gaat, verschijnt onderaan in de Arduino IDE de melding **“Done compiling”**. De Arduino IDE heeft je sketch gecompileerd tot een programma dat het board kan uitvoeren.

Klik vervolgens op de **Upload**-knop. Het board wordt gereset en de gecompileerde sketch wordt naar het board gestuurd en in het geheugen opgeslagen. Daarna voert het board het programma uit.

Als alles goed verlopen is, verschijnen de meldingen **“Done compiling”** en **“Done uploading”**. Dat betekent dat je sketch succesvol gecompileerd en geüpload is.

> **Tip**

> Let op de RX- en TX-LED's op het board. Die knipperen wanneer data wordt verzonden of ontvangen. Tijdens het uploaden zullen ze dus flikkeren. Gebeurt dat niet, of verschijnt er een foutmelding in plaats van “Done uploading”, dan is er mogelijk een communicatieprobleem tussen je computer en Arduino. Controleer of je de juiste serial port en het juiste Arduino-model hebt geselecteerd.

Zodra je sketch op het Arduino board is geüpload, blijft die daar opgeslagen totdat je een andere sketch uploadt.

Als de sketch correct is geüpload, brandt de LED één seconde en is ze vervolgens één seconde uit.

### 🔎 Een blik op de code

```
const int ledPin = 13;
```

Hier geven we aan dat de LED die we laten knipperen, aangesloten is op pin 13 van de Arduino. We definiëren een variabele van het **type** `int`, met de **naam** `ledPin` en de **waarde 13**.

`const` staat voor constant. Het geeft aan dat de variabele niet meer gewijzigd mag worden nadat ze een waarde heeft gekregen. De waarde wordt dus vastgelegd. Het is niet verplicht: `int ledPin = 13;` werkt ook.

Je kunt ook `#define ledPin 13` gebruiken. Hiermee geef je aan dat de preprocessor het woord `ledPin` vóór het compileren vervangt door het getal 13.

**Arduino verwacht altijd dat twee functies aanwezig zijn:** `setup()` en `loop()`.

De functie `setup()` wordt één keer uitgevoerd aan het begin van het programma.

De functie `loop()` wordt tijdens de uitvoering van het programma telkens opnieuw aangeroepen.

`pinMode()` vertelt Arduino hoe een bepaalde pin moet worden ingesteld. Digital pins kunnen zowel als INPUT als OUTPUT worden gebruikt.

`digitalWrite()` stelt een pin die als `OUTPUT` is ingesteld in op HIGH of LOW. Het eerste argument bepaalt welke pin je aanstuurt, het tweede argument bepaalt de toestand van de pin. HIGH komt op de Arduino UNO overeen met ongeveer 5V en LOW met ongeveer 0V. Hierdoor gaat de aangesloten LED branden of uit.

`delay()` wacht het opgegeven aantal milliseconden.

Commentaar schrijven in Arduino: tekst die begint met `//` wordt genegeerd. Voor een blok commentaar kun je ook `/* ... */` gebruiken.

## 2.2. Lingua Arduino

De Arduino-programmeertaal bestaat uit drie grote onderdelen: **functions**, **variables en constants** (values) en **structure**.

Bekijk de reference page voor de documentatie van de verschillende onderdelen van de Arduino-taal.

## 2.3. Speel met de parameters

Pas de waarden van de twee `delay()`-tijden aan en ontdek hoe het knipperritme verandert. Vergeet niet om je sketch opnieuw te compileren en te uploaden nadat je wijzigingen hebt aangebracht.

## 2.4. De Autocomplete-feature van IDE 2.x

Autocomplete helpt je sneller code te schrijven en al doende te leren.

Om autocompletion in of uit te schakelen, ga je naar **File > Preferences** en schakel je **Editor Quick Suggestions** in of uit.

Om autocompletion te gebruiken, moet je eerst een board selecteren. Daarna verschijnen de suggesties volgens de documentatie automatisch wanneer je begint te typen.

Dat gebeurt echter niet altijd. Soms moet je de suggesties zelf oproepen met **Ctrl+Space** (ook op Mac, afhankelijk van je systeeminstellingen).

## 2.5. Andere actuators

Je kunt de LED ook vervangen door andere actuators, bijvoorbeeld:

- Een **buzzer** of beeper is een klein apparaat dat een zoemend geluid maakt en gebruikt wordt om signalen te geven.
- Een **relay** (relais) is een elektrisch aangestuurde schakelaar. Met een elektrisch signaal kan een relay een andere elektrische schakeling in- of uitschakelen, bijvoorbeeld om verlichting of andere elektrische apparaten te bedienen.
- Een **electromagnet** (elektromagneet) is een magneet die ontstaat wanneer er elektrische stroom door een spoel loopt. Een *solenoid* is een specifieke toepassing waarbij een elektromagneet een lineaire beweging kan veroorzaken.
- ...

Maar daarvoor moeten we eerst weten hoe we ze aansluiten.

# 3. Circuits bouwen: een LED aansturen

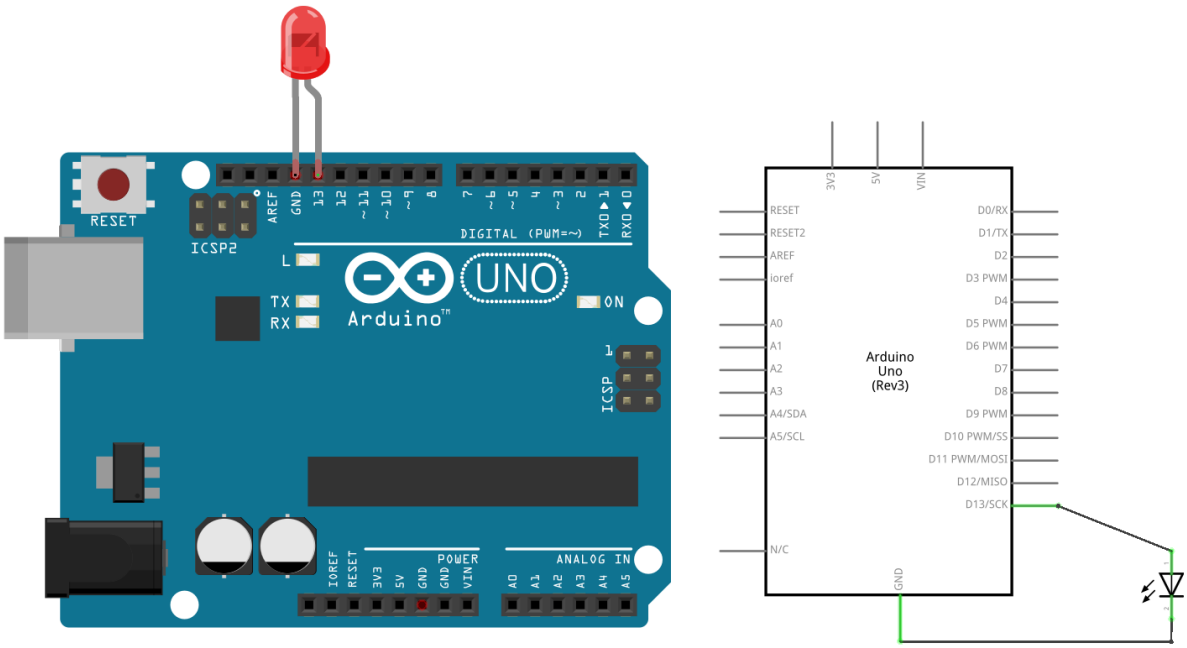
## 3.1. Wiring diagrams & schematics

We willen nu een externe LED op het Arduino board aansluiten. Ik kan stap voor stap uitleggen hoe je de verbindingen maakt, maar zou het niet veel eenvoudiger zijn om een diagram te tekenen waarop je ziet hoe de componenten met elkaar verbonden zijn? De anode (het langste pootje) van de LED verbinden we via een resistor met pin 13 van de Arduino UNO. De cathode (het kortste pootje) verbinden we met ground.

Het kunnen lezen van deze **diagrammen** (bedradingsschema's) is een belangrijk onderdeel van het bouwen van circuits. **Schematics** (elektronische schema's) gebruiken gestandaardiseerde symbolen waarmee je elektronische schakelingen kunt begrijpen en bouwen.

Elk elektronisch component heeft een schematisch symbool. Die symbolen worden in verschillende programma's samengebracht tot een schematic, maar je kunt ze ook met de hand tekenen. Wil je je verder verdiepen in elektronica en het bouwen van circuits, dan is het leren lezen van schematics een belangrijke stap.

Hieronder zie je links de schematic van de bovenstaande schakeling en rechts een meer visueel wiring diagram, gemaakt met Fritzing. In deze tutorial gebruiken we vooral dit soort wiring diagrams.




*Om onze pin en LED te beschermen moet eigenlijk hier nog een resistor tussen. We zien straks hoe we dat doen.*

> **Tip**
>
> Bekijk eventueel ook deze uitgebreidere tutorial: How to Read a Schematic.

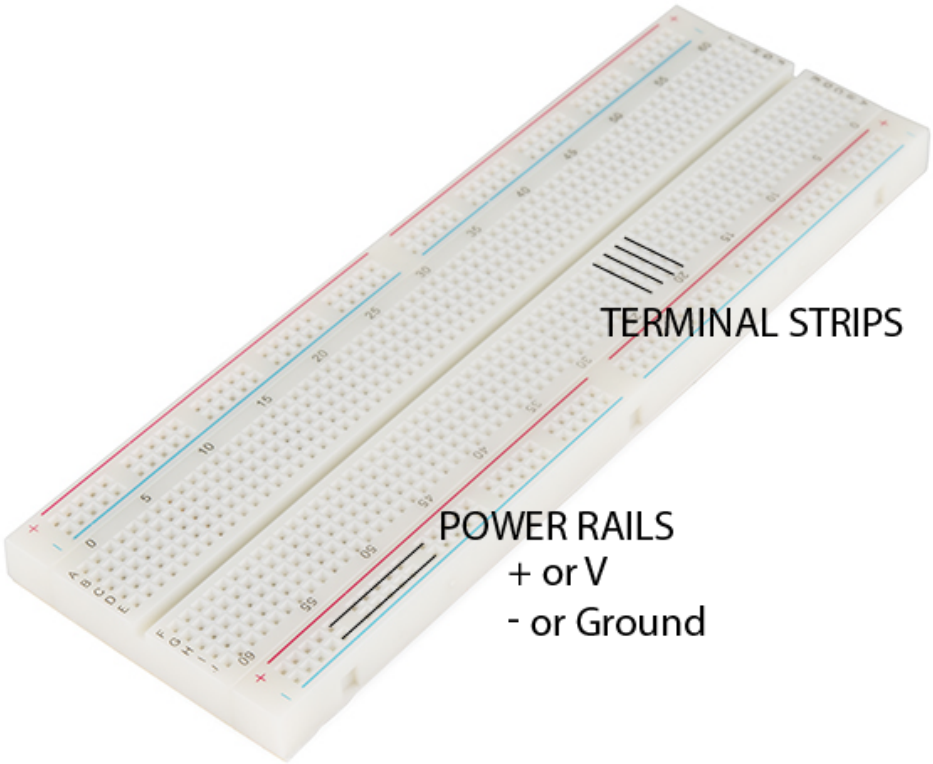
## 3.2. Componenten

Laten we aan de slag gaan met physical computing en elektronische componenten die we kunnen aansluiten en aansturen.

Om verbindingen te maken zonder te solderen, komt een *solderless* breadboard goed van pas.

### Breadboard

Een breadboard is een kunststof board met een heleboel gaatjes. Onder elk gaatje zit een metalen contact met een klemsysteem. Wanneer je de pootjes van een component in de gaatjes steekt, maken ze elektrisch contact met andere gaatjes die intern met elkaar verbonden zijn. Veel breadboards hebben ook *power rails* voor de stroomvoorziening, waardoor je circuits eenvoudiger kunt opbouwen.




Steek vervolgens de LED in het breadboard en verbind pin 13 en GND met de Arduino. Sluit de anode (het langste pootje) via een resistor (bijvoorbeeld 220 Ω of 330 Ω) aan op pin 13 en de cathode (het kortste pootje) op GND.

> **Tip**
>
> Wil je wat extra uitleg en ondersteuning, bekijk dan deze uitstekende Sparkfun-pagina: How to Use a Breadboard.

## Draad

Draden gebruik je om componenten met elkaar te verbinden. Ze bestaan in verschillende types en diktes. Twee veelgebruikte types zijn *solid core* (een massieve draad) en *stranded wire* (een soepele draad opgebouwd uit meerdere dunne draadjes). Op het breadboard gebruiken we *jumper wires*, ook wel *jumper leads* genoemd.

## Schakelaars of knoppen

Schakelaars kunnen een elektrische verbinding onderbreken of herstellen. Je kunt draden aansluiten op de contacten van een schakelaar. Door de schakelaar te bedienen, wordt de elektrische verbinding geopend of gesloten.

Er bestaan verschillende soorten schakelaars, waaronder *toggle switches* en *momentary switches*. Een *toggle switch* blijft in de gekozen positie staan, terwijl een *momentary switch* (of *pushbutton*) terugkeert naar zijn oorspronkelijke positie zodra je hem loslaat. In deze tutorial gebruiken we die laatste.

## Light-emitting diodes (LED)

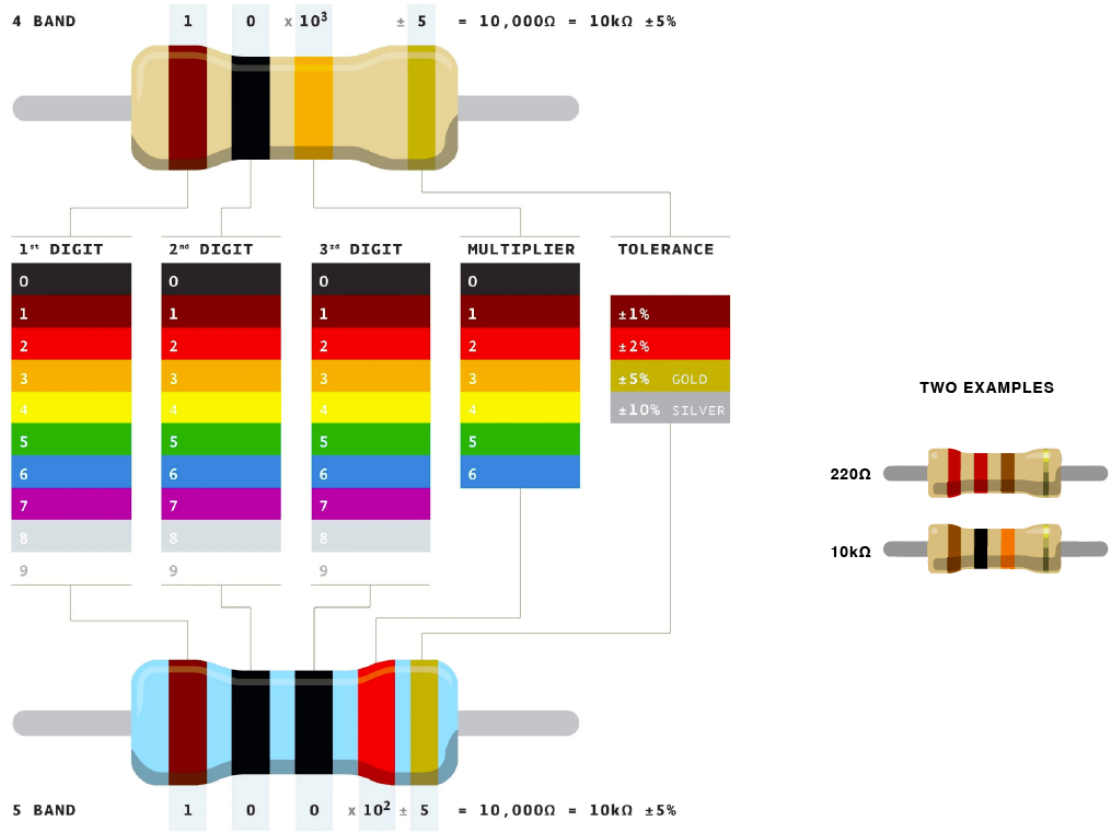
LED's zijn een van de meest gebruikte outputs van een microcontroller. Ze verbruiken weinig stroom en kunnen eenvoudig worden aangestuurd. Een LED (*light-emitting diode*) is een diode die licht uitstraalt. Om te begrijpen hoe een LED werkt, moeten we eerst weten hoe een diode werkt.

Een diode is als een eenrichtingsstraat: ze laat elektrische stroom slechts in één richting door. Diodes zijn dus gepolariseerd. De twee aansluitingen worden de cathode (–) en anode (+) genoemd.

## Resistors

Resistors (weerstanden) beperken de elektrische stroom in een circuit. Daarbij wordt een deel van de elektrische energie omgezet in warmte. Ze hebben twee aansluitingen en geen polariteit.

De waarde van een resistor wordt uitgedrukt in ohm (Ω). Die waarde bepaalt hoeveel weerstand de resistor biedt aan de elektrische stroom. Hieronder leer je de kleurcodes lezen.




## Potentiometers

Potentiometers zijn variabele resistors met drie aansluitingen. De weerstand verandert afhankelijk van de positie van het draaiende of schuivende contact, de *wiper*. De wiper zit op het middelste pootje en levert een variabele spanning. Je gebruikt een potentiometer met Arduino als *voltage divider*. Dat betekent dat je alle drie de contacten aansluit: één op GND, één op een analog input pin en één op +5V (of 3V3).

Andere veelgebruikte sensoren waarvan de weerstand verandert, zijn photocells (LDR's), thermistors, force-sensitive resistors (FSR's) en bend sensoren. Deze componenten hebben doorgaans twee aansluitingen. Om ze correct uit te lezen, moeten we een tweede resistor toevoegen. Daar komen we later op terug.

> **Tip**
>
> Bekijk ook deze uitstekende website als je meer wilt leren over de basisprincipes van elektriciteit of je verder wilt verdiepen in physical computing.

# 4. Inputs: Arduino laten reageren

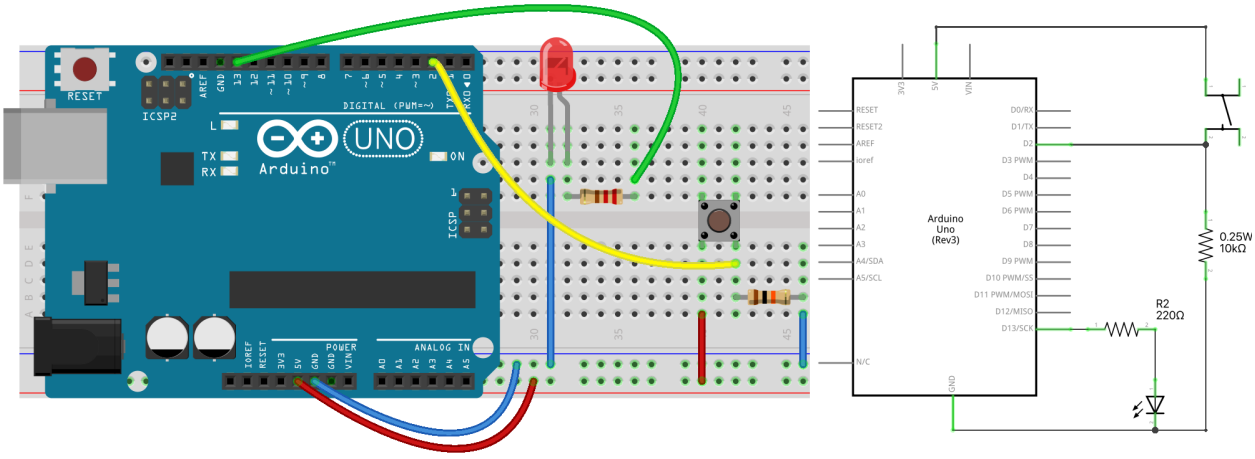
## 4.1. Druk op de knop

In ons eerste voorbeeld was de LED onze actuator en stuurden we die aan met Arduino. Nu willen we de LED laten reageren op een externe input, bijvoorbeeld wanneer we op een knop drukken. Daarvoor hebben we **een sensor** nodig. Een pushbutton is een van de eenvoudigste inputs die we kunnen gebruiken.

Laten we eerst het circuit maken.

### Circuit

- LED aangesloten via een resistor tussen pin 13 en GND
- Pushbutton aangesloten tussen pin 2 en +5V
- 10 kΩ resistor aangesloten tussen pin 2 en GND (pull-down resistor)




### Code

```
 // constants don't change:
const int buttonPin = 2;
const int ledPin =  13;

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

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

Als alles correct is aangesloten, gaat de LED branden wanneer je de button indrukt. Gelukt? Mooi!

### 🔎 Een blik op de code

Hier maken we kennis met de functie `digitalRead()` en de `if` - en `else` -instructies. Die zijn essentieel in programming: ze laten een computer beslissingen nemen.

`digitalRead()` leest de logische toestand van een digital pin: HIGH of LOW. Op een Arduino UNO komt HIGH overeen met een hoge spanning (doorgaans 5V) en LOW met een lage spanning (doorgaans 0V).

> **Tip**
>
> Let op het verschil tussen `==` en `=`. De eerste operator vergelijkt twee waarden en levert `true` of `false` op. De tweede kent een waarde toe aan een variabele.

### 🔎 En nog enkele woorden over het circuit

We hebben een verbinding met ground nodig via een resistor. Wanneer de button ingedrukt is, wordt pin 2 verbonden met +5V en leest Arduino HIGH. Maar wanneer de switch open staat, is de digital input pin niet verbonden met een vaste spanning. De pin zweeft (*floats*) en kan daardoor onvoorspelbare waarden uitlezen. Dit lossen we eenvoudig op met een **pull-down resistor**, die we tussen de pin en GND plaatsen.

> **Tip**
>
> Wil je je elektronische schakeling zo eenvoudig mogelijk houden, dan kun je `INPUT_PULLUP` gebruiken in de `pinMode()` -functie. Daarmee activeer je de interne **pull-up resistor** van de microcontroller. Die zorgt ervoor dat de pin standaard HIGH leest. Wanneer je de button indrukt en de pin met GND verbindt, wordt de waarde LOW. De logica is dus omgekeerd.

Ook het circuit verandert: één aansluiting van de pushbutton verbind je met GND, de andere met een digital pin. De externe resistor is niet meer nodig. Bekijk dit voorbeeld.

## 4.2. Een on/off-button die vast blijft staan

Je vinger op de button houden zolang je licht nodig hebt, is niet erg praktisch.

Laten we **een ander gedrag** programmeren, zodat de LED blijft branden nadat we de button loslaten. Daarvoor moeten we een vorm van *geheugen* implementeren: een softwaremechanisme dat onthoudt wanneer we op de button hebben gedrukt.

```
/* Turn on LED when the button is pressed
  and keep it on after it is released */

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

int val = 0;          // val will be used to store the state of the input pin
int old_val = 0;      // this variable stores the previous value of "val"
int buttonState = 0;  // variable that will store the pushbutton status

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

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

  // check if there was a transition
  if ((val == HIGH) && (old_val == LOW)) {
    buttonState = 1 - buttonState;
    delay(10);    // small delay for debouncing
  }

  old_val = val;  // val is now old, let's store it

  // check if the pushbutton is pressed. If it is, the buttonState is HIGH:
  if (buttonState == 1) {
    digitalWrite(ledPin, HIGH); // turn LED on
  } else {
    digitalWrite(ledPin, LOW);  // turn LED off
  }
}
```

> ⚠ **Warning**
>
> 🦊 Wacht even! Wat is **debouncing**? In deze tutorial wordt het uitgelegd en geïllustreerd, inclusief een betere methode die het programma niet blokkeert.

## 4.3. Andere on/off-sensoren

Nu je geleerd hebt hoe je een pushbutton gebruikt, weet je ook dat er andere eenvoudige sensoren zijn die volgens hetzelfde *on/off*-principe werken, zoals:

- **Switches** (schakelaars) blijven in hun gekozen positie staan, in tegenstelling tot momentary buttons.
- **Thermostaten** schakelen een circuit in of uit wanneer de temperatuur een bepaalde waarde bereikt.
- **Magnetic switches** (reed switches) hebben contacten die met elkaar verbonden worden wanneer er een magneet in de buurt komt.
- **Carpet switches** zijn schakelaars die je onder een tapijt of deurmat kunt leggen om te detecteren of iemand erop staat (of een zware kat).
- **PIR-sensor** (*Passive Infrared Sensor*). Deze sensor detecteert veranderingen in infraroodstraling, bijvoorbeeld wanneer een persoon of een ander warm lichaam in de buurt beweegt.
- **Tilt switches** zijn elektronische schakelaars met twee contacten en een klein metalen balletje. Wanneer de schakelaar kantelt, verandert de positie van het balletje en wordt het circuit geopend of gesloten.

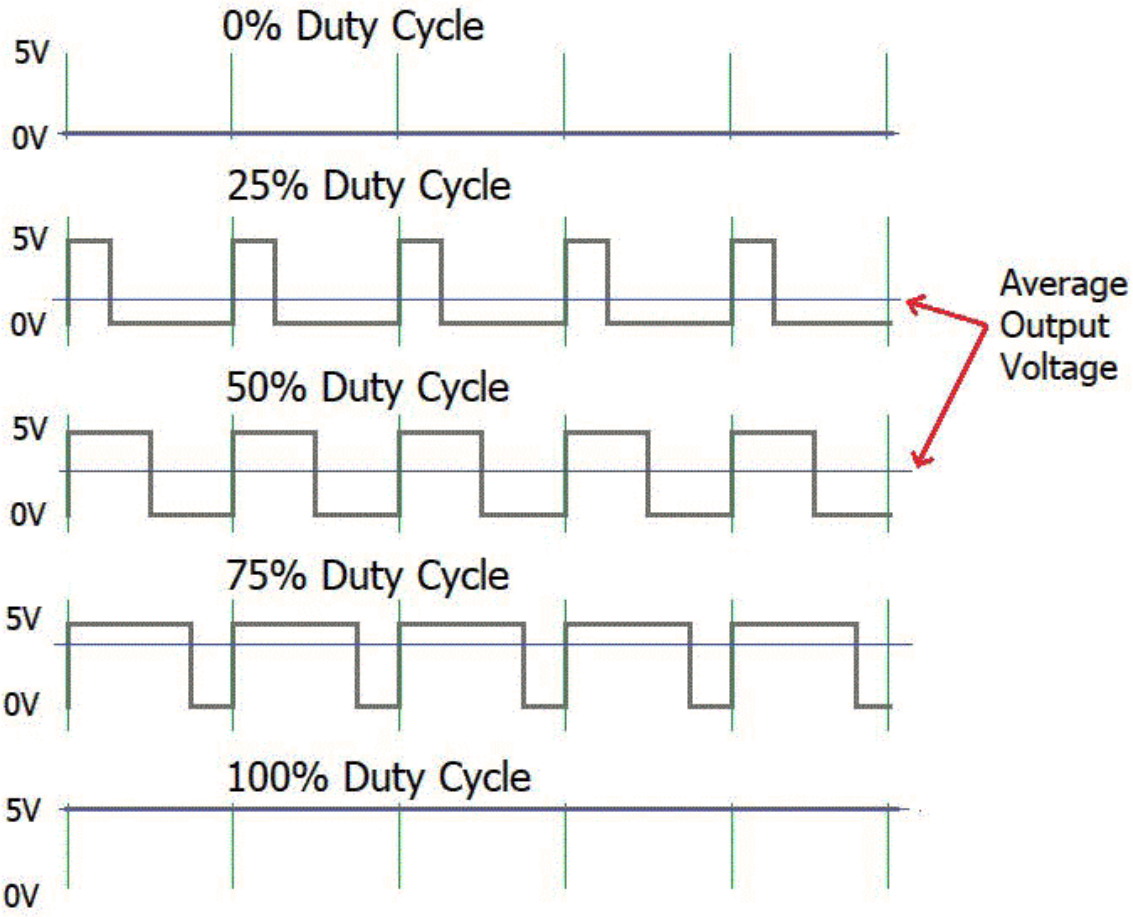
# 5. Variabele output: PWM en servo motoren

## 5.1. Een LED dimmen met PWM

**PWM** (*Pulse Width Modulation*) is een techniek waarmee je een digitaal signaal zo kunt moduleren dat het zich in bepaalde toepassingen gedraagt als een analoog signaal. De Arduino UNO kan geen echte analoge spanning uitsturen via zijn digital pins, maar wel digitale signalen schakelen tussen LOW (0V) en HIGH (5V).

We gebruiken PWM om bijvoorbeeld LED's te dimmen, motors aan te sturen, geluid te synthetiseren en nog veel meer. PWM maakt dit mogelijk doordat we **de tijdsduur waarin het signaal HIGH is, kunnen variëren. Daardoor kan het signaal zich in bepaalde toepassingen gedragen als een analoge output**.

Het signaal is op elk moment HIGH (5V) of LOW (0V). Binnen een vaste tijdsperiode kunnen we de verhouding tussen de tijd dat het signaal HIGH en LOW is aanpassen. Dat noemen we de *duty cycle*.

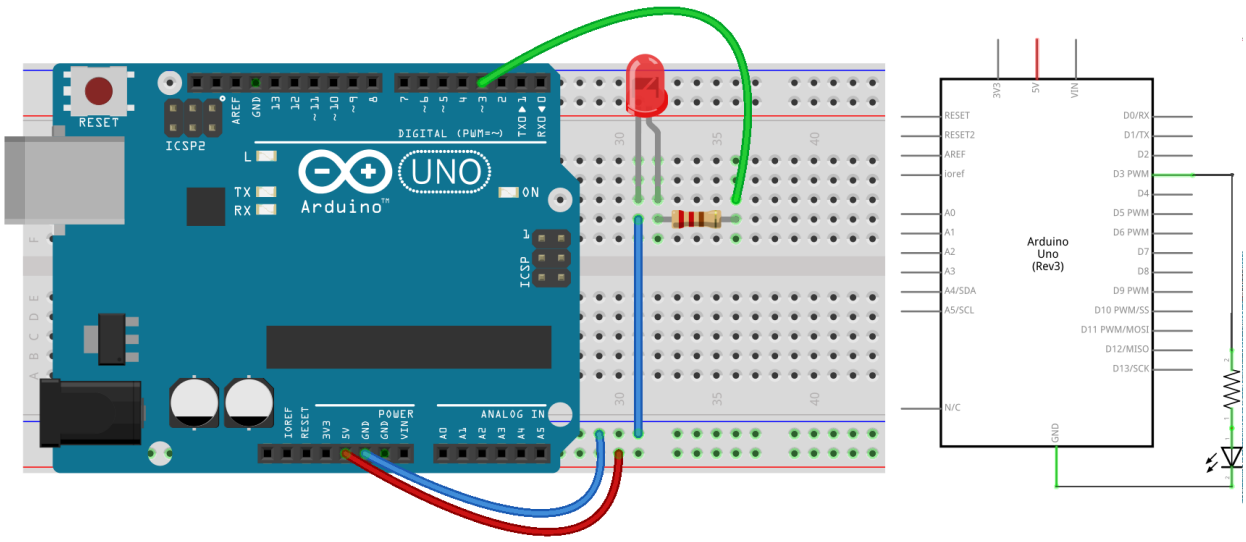



- **Pulse width**: de tijdsduur waarin het signaal HIGH is.
- **Pulse period**: de totale duur van één cyclus (HIGH en LOW samen).
- **Voltage level**: de spanning van het signaal (bijvoorbeeld 0V–5V).

Een Arduino UNO heeft 6 PWM-pins: digital pins 3, 5, 6, 9, 10 en 11. Ze zijn allemaal aangeduid met een `~` (tilde).

We onderzoeken PWM door de brightness van een LED geleidelijk te veranderen.

### Circuit




## Code

```
int ledPin = 3;      // LED connected to digital pin 3
int fadeAmount = 5;  // how many points to fade the LED by


void setup() {
  // nothing happens in setup
}

void loop() {
  // fade in from min to max in increments of fadeAmount points
  for (int fadeValue = 0 ; fadeValue <= 255; fadeValue += fadeAmount) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }

  // fade out from max to min in increments of fadeAmount points
  for (int fadeValue = 255 ; fadeValue >= 0; fadeValue -= fadeAmount) {
    // sets the value (range from 0 to 255):
    analogWrite(ledPin, fadeValue);
    // wait for 30 milliseconds to see the dimming effect
    delay(30);
  }
}
```

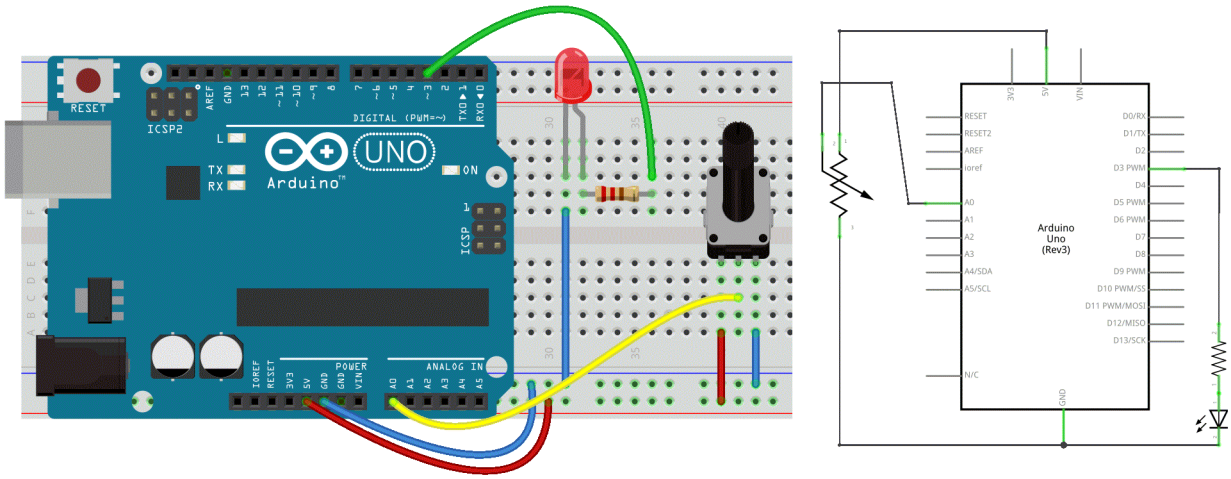
# 5.2. Input naar output

**Laten we nu input en output met elkaar verbinden.**

In een vorig experiment hebben we een LED aangestuurd met een pushbutton. Nu gebruiken we een potentiometer om de brightness van de LED te regelen.

Arduino leest de analoge waarde van de potentiometer en zet die om naar een PWM-output. Omdat `analogWrite()` op de Arduino UNO waarden tussen 0 en 255 gebruikt, moeten we de input omzetten van een waarde tussen 0 en 1023 naar een waarde tussen 0 en 255.

## Circuit




## Code

```
/* Set the brightness of ledPin to a brightness specified by the
  value of the analog input */

const int ledPin = 3;      // LED connected to digital pin 3
const int analogPin = A0;  // potentiometer connected to analog pin 0

int val = 0;               // variable to store the read value
int ledVal;                // variable to store the output value


void setup() {
  // Nothing here, as analog pins are automatically set as inputs &
  // it is not needed to set the pin as an output before calling analogWrite()
}
void loop() {
  // read the value from the sensor
  val = analogRead(analogPin);
  // turn the ledpin on at the brightness set by the sensor
  // Map the values to a range between 0 and 255
  // because analogWrite() uses this range
  ledVal = map(val, 0, 1023, 0, 255);
  analogWrite(ledPin, ledVal);
  delay(10);
}
```

Na het uploaden van het programma zie je de brightness van de LED veranderen wanneer je aan de potentiometer draait.

## 5.3. Servo motor control

We vervangen nu de LED door een **servo motor**.

Servos zijn motoren met een shaft (as) die naar een specifieke positie kan draaien. Ze hebben meestal een bereik van 0 tot 180 graden. Met Arduino kun je een servo naar een bepaalde positie sturen. In dit onderdeel bekijken we hoe je een servo aansluit en hoe je de positie ervan regelt met een potentiometer.

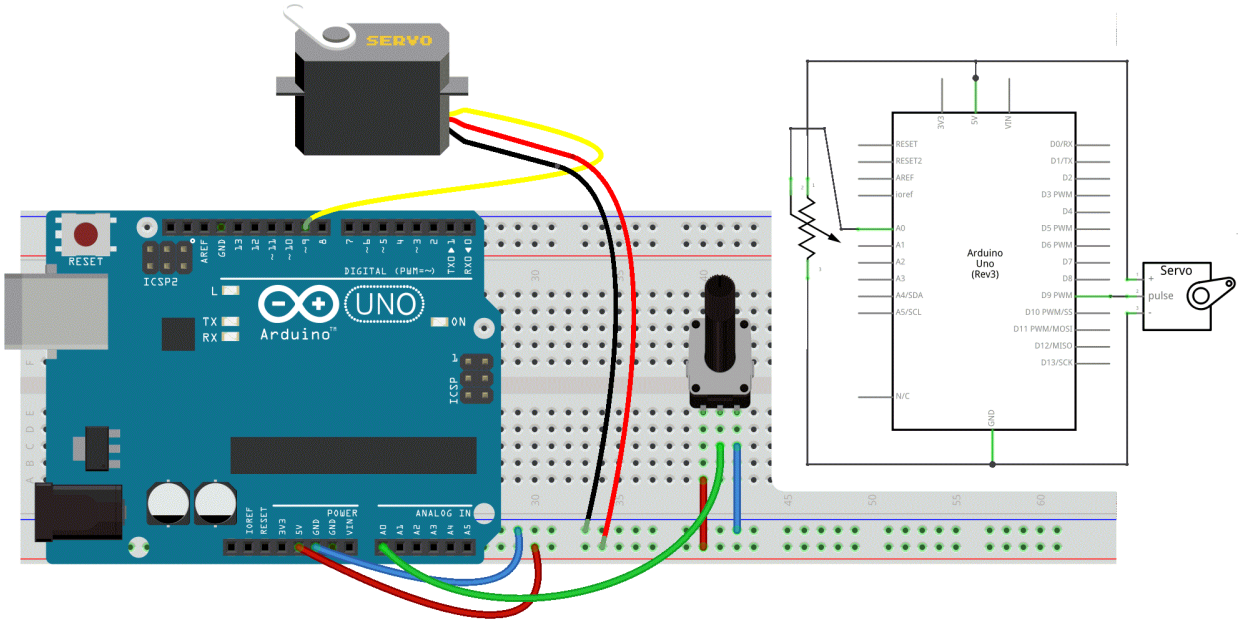
Een servo motor bestaat uit verschillende onderdelen: een motor, een gearbox en een feedback circuit. Hij heeft doorgaans drie aansluitingen: power, ground en een control pin.

*Er bestaat ook een ander type servo dat continu kan draaien en waarbij je de snelheid kunt regelen. Dit type is bijzonder handig voor kleine voertuigen met gears.*

## Circuit

Onze servo motor heeft een female connector met drie pins.

- De donkerste draad, hier bruin, is meestal de ground. Verbind die met Arduino GND.
- De power wire is doorgaans rood. Verbind die met 5V op Arduino.
- Verbind de resterende draad van de servo met digital pin 9 op Arduino.




*Hou er rekening mee dat servos behoorlijk wat stroom kunnen verbruiken. Wanneer je meerdere servos aanstuurt, heb je waarschijnlijk een aparte power supply nodig. Gebruik daarvoor niet de 5V-pin van je Arduino. Vergeet ook niet de ground van de externe voeding met de GND van Arduino te verbinden.*

## Code

In dit voorbeeld gebruiken we **een library** die het programmeren een stuk eenvoudiger maakt.

Net als bij andere programmeerplatformen kun je de Arduino-software uitbreiden met libraries. Die bieden extra functionaliteit en maken het programmeren eenvoudiger, omdat ze code bevatten waarmee je bepaalde modules, sensoren enzovoort kunt aansturen. Een aantal libraries is standaard geïnstalleerd in de IDE, maar je kunt ook andere libraries downloaden of zelf maken.

Om een library in je sketch te gebruiken, selecteer je die via **Sketch > Include Library** of voeg je de library toe met de opdracht `#include <name_of_library>` .

```
#include <Servo.h>

Servo myservo;    // create servo object to control a servo

int potpin = A0;  // analog pin used to connect the potentiometer
int val;          // variable to read the value from the analog pin

void setup() {
  myservo.attach(9);  // attaches the servo on pin 9 to the servo object
}

void loop() {
  // read the value of the potentiometer
  val = analogRead(potpin);
  // scale it to use it with the servo (value between 0 and 180)
  val = map(val, 0, 1023, 0, 180);
  // sets the servo position according to the scaled value
  myservo.write(val);
  // waits for the servo to get there
  delay(15);
}
```

Controleer altijd de mogelijkheden en beperkingen van een library op de bijbehorende reference page. Dit is de reference page van Servo.

De belangrijkste functies die we hier gebruiken zijn:

- `Servo objectname;` maakt een servo-object aan.
- `objectname.attach(pin)` koppelt de servo aan een digital pin.
- `objectname.write(angle)` bepaalt de positie van de servo (0 tot 180 graden).

# 6. Additional Learning resources

- De onderdelen Learn en Tutorials op de Arduino-website helpen je op weg.
- Bekijk ook deze website als je meer wilt leren over de basisprincipes van elektriciteit of je verder wilt verdiepen in physical computing.

Dat was het voorlopig.
👋🏻