2. Hello World!

Een **"Hello World!"** in de Arduino-wereld is een knipperende LED.
Je hebt enkel een Arduino-board en een USB-kabel nodig.
Open een nieuue sketch in de IDE. De onderstaande regels code zijn al geschreven. Ze vormen de basis van elk programma. Later meer hierover.

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

}

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

}
```

Geef de Sketch een naam en sla het op.
Typ vervolgens de volgende tekst in de Editor, maar je kunt de regels die beginnen met `//` overslaan, omdat dit comments zijn.

```
// LED connected to digital pin 13
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);
}
```

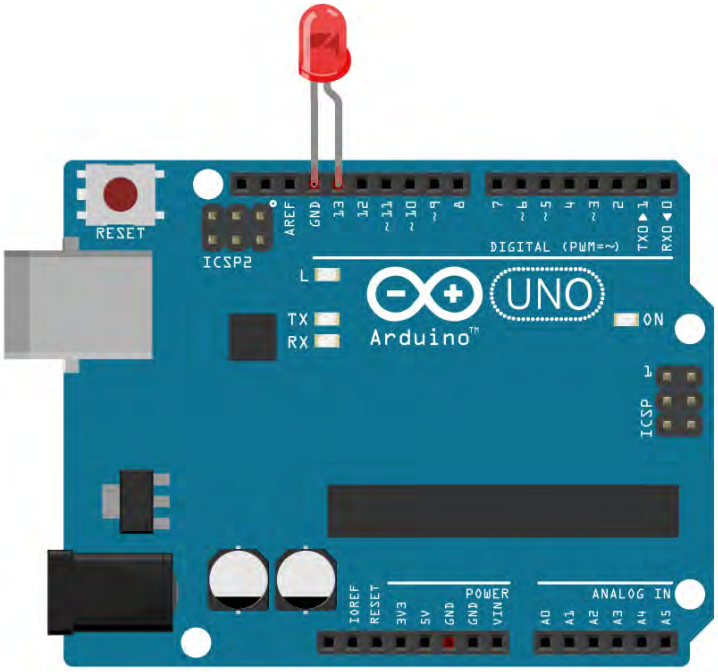
Druk op de **Verify**-knop om te controleren of je code correct is.
Als alles goed is, zie je het bericht **“Done compiling”** onderaan in de Arduino IDE verschijnen. De Arduino IDE heeft je sketch vertaald naar een uitvoerbaar programma dat door het board kan worden uitgevoerd.

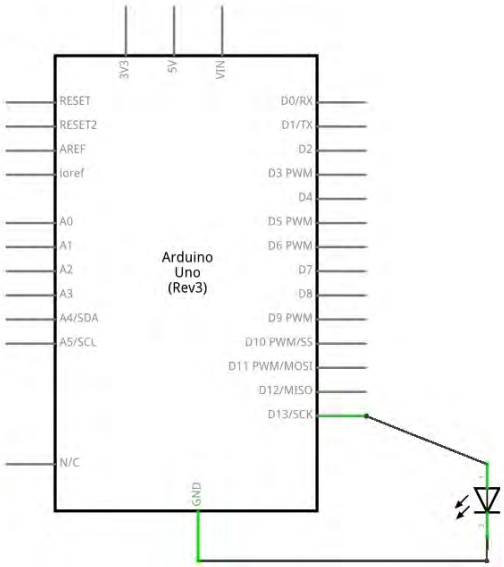
Druk nu op de **Upload**-knop. Dit zal het board resetten en dwingen om zijn huidige functies te stoppen. Vervolgens wordt de gecompileerde sketch naar het board gestuurd en in het geheugen opgeslagen. Daarna zal het board deze uitvoeren.

Als alles goed is gegaan, zie je de meldingen **“Done compiling.”** en **“Done uploading.”** verschijnen om je te laten weten dat het proces correct is voltooid.

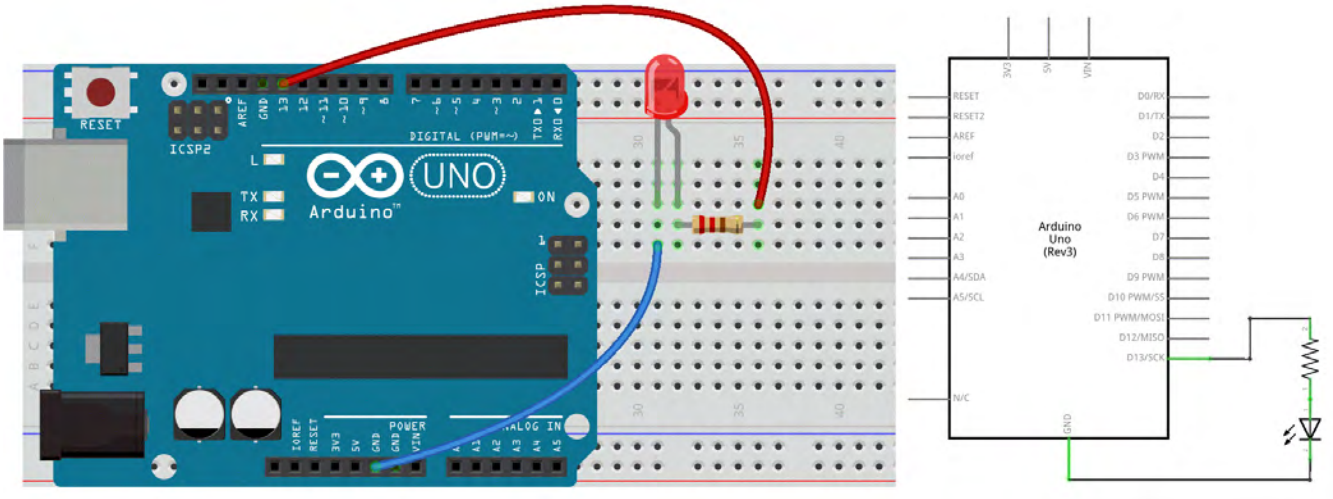
Je kunt de waarden van de 2 vertragingstijden aanpassen om veranderingen in het knipperritme te zien. Vergeet niet om de code opnieuw te compileren en te uploaden nadat je wijzigingen hebt aangebracht.

3. Een Externe LED





3b. Een LED op een Breadboard



Laten we nu enkele andere actuatoren proberen:

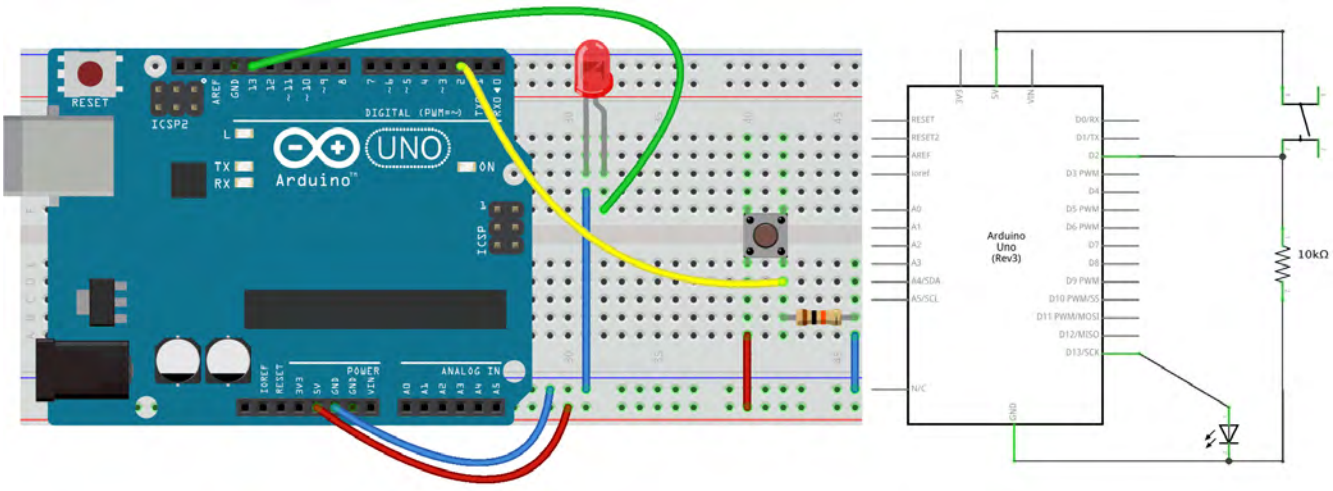
- Een zoemer of buzzer is een klein apparaat dat een zoemend geluid maakt en wordt gebruikt voor signalering.
- Een **relais_** is een elektrisch bediende schakelaar. Het gebruikt een laagspanningsbesturingssignaal om meestal een hogere spanning te schakelen. Het kan ook worden gebruikt om verlichting, elektrische en andere apparatuur te bedienen.
- ...

4. Push the Button

In ons eerste voorbeeld was de LED onze actuator, en onze Arduino bestuurde deze. Als we ons voorstellen dat een externe parameter deze LED moet bedienen — onze vinger — dan hebben we **een sensor** nodig. En de eenvoudigste vorm van sensor die beschikbaar is, is een **pushbutton**.

Circuit

- LED aangesloten van pin 13 naar ground
- pushbutton aangesloten op pin 2 vanaf +5V
- 10K weerstand of resistor aangesloten op pin 2 naar ground



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);
  }
}
```

b. Sticky On/Off Button

Laten we een tweede functionaliteit programmeren, zodat de knop werkt als een schakelaar die ingedrukt blijft.

Code

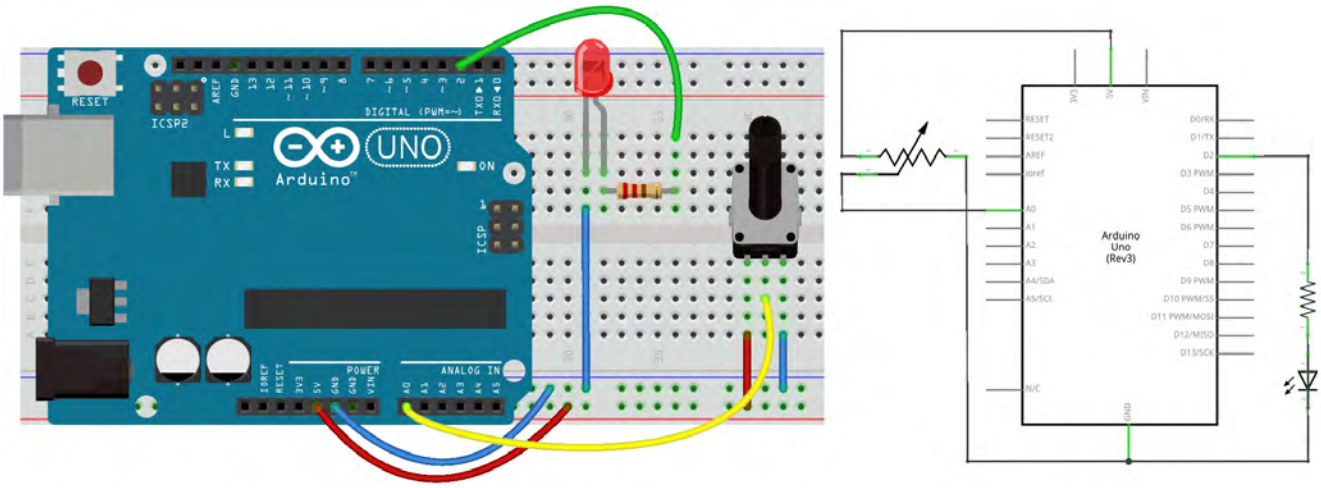
```
/* Turn on LED when the button is pressed
  and keep it on after it is released */

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
  }
}
```

5a. Analoge Sensoren

De volgende sketch en elektronisch schema demonstreren analog input door een analoge sensor (een potentiometer of trimpot) uit te lezen op analog pin 0 en een LED aan en uit te schakelen die is verbonden met digital pin 2. De tijd dat de LED aan of uit is, hangt af van de waarde die wordt verkregen via `analogRead()`.

Circuit

- potentiometer: de middelste pin van de potentiometer naar analog input 0, één zijpin (maakt niet uit welke) naar ground, de andere zijpin naar +5V
- LED: een 220Ω resistor verbindt digital output 2 met de anode (het lange been) van de LED, de cathode (het korte been) is verbonden met ground.
  Eigenlijk kan de resistor ook tussen de cathode en ground worden geplaatst, omdat in een serieschakeling de volgorde van de componenten niet uitmaakt: de stroom moet door alle onderdelen!



Code

```
int sensorPin = A0;   // select the input pin for the potentiometer
int ledPin = 2;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT
  pinMode(ledPin, OUTPUT);
  // there is no need to set our analog in pin
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

5b. talk2me

Laten we een **seriële verbinding** opzetten tussen de Arduino en de computer om de veranderende waarden te volgen.
In de onderstaande code zetten we de waarden van 0–1023 om naar een aangepast bereik van 10–500. Beide variabelen worden via de serial port uitgestuurd waardoor we ze kunnen bekijken in de Arduino Serial Monitor.

Om dat te doen pas je de code aan zoals hieronder.
Klik vervolgens op de Serial Monitor-knop in de toolbar en kies dezelfde baud rate die is ingesteld in de `Serial.begin()`-functie.

Het circuit blijft hetzelfde.

Code

```
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 2;        // select the pin for the LED
int sensorValue = 0;   // variable to store the value coming from the sensor
int outputValue = 0;   // variable to store a scaled value of the sensorvalue

void setup() {
  // declare the ledPin as an OUTPUT
  pinMode(ledPin, OUTPUT);
  // initialize serial communications at 9600 bps
  Serial.begin(9600);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);

  // map or scale it to a custom range:
  outputValue = map(sensorValue, 0, 1023, 10, 500);

  // print the results to the Serial Monitor:
  Serial.print("sensor = ");
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

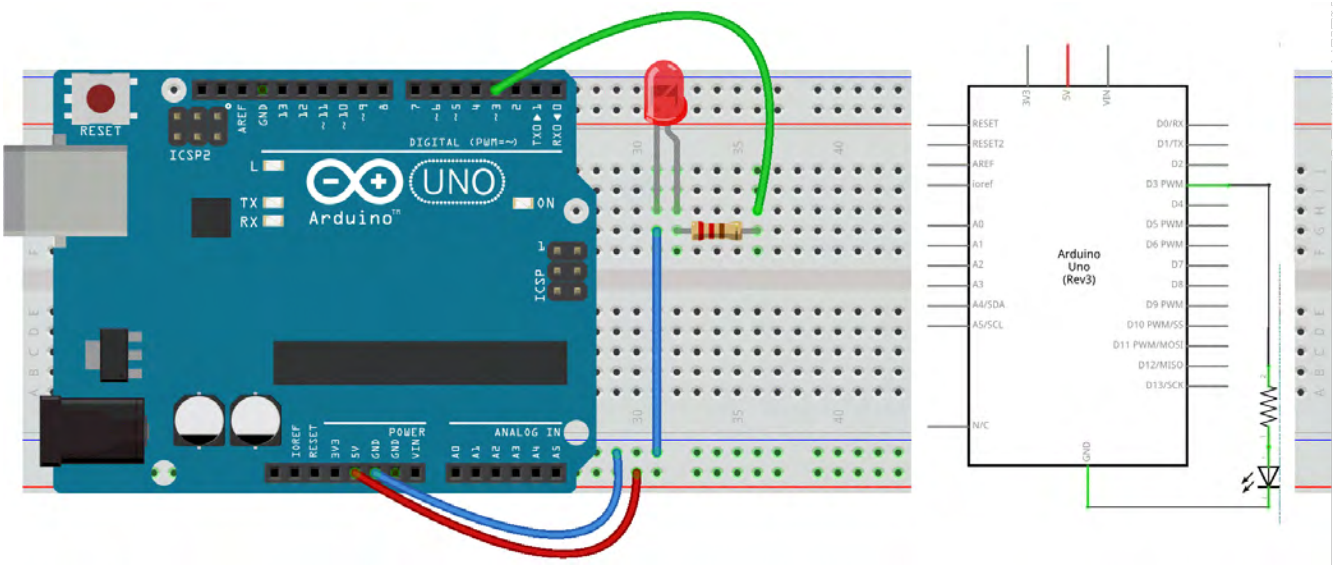
10a. Analog Out, a fading LED

PWM, kort voor **Pulse Width Modulation**, is een techniek waarmee een analoog signaalniveau wordt gecodeerd in een digitaal signaal.

Op een Arduino Uno zijn er 6 PWM-interfaces: digital pins 3, 5, 6, 9, 10 en 11. Deze worden aangeduid met een ~ (tilde) op de printplaat.

We gaan deze PWM-techniek verkennen door de helderheid van een LED in de loop van de tijd te veranderen.

Circuit



Code

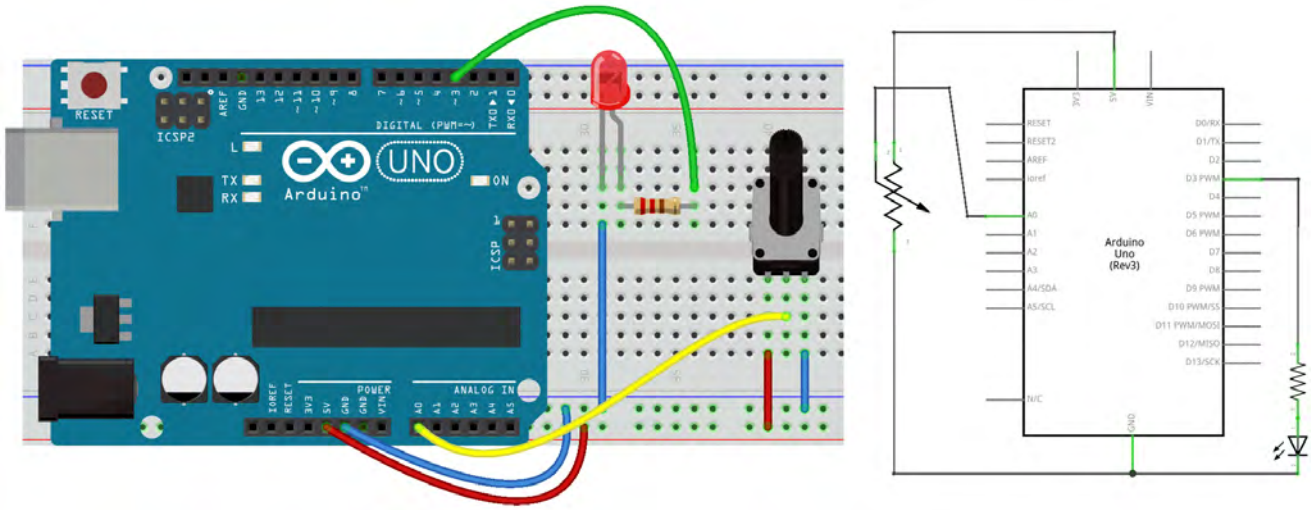
```
int ledPin = 3;      // LED connected to digital pin 3
int fadeAmount = 5;  // how many steps to fade the LED by


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

void loop() {
  // fade in from min to max in increments of ? 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 ? 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);
  }
}
```

6b. Analog Input 2 Output

Nu gaan we onze Input koppelen aan de Output.
In een eerder experiment hebben we een button-controlled LED gemaakt, waarbij een digitale knop een digitale pin aanstuurt. Nu gaan we een potentiometer gebruiken om de helderheid van een LED te regelen.

De Arduino leest de analoge waarde van de potentiometer en gebruikt deze om een PWM-signaal naar de LED te sturen. Omdat PWM alleen werkt in het bereik van 0–255, moeten we de invoer van onze sensor omzetten van 0–1023 naar 0–255. Dit doen we met de functie `map()`.

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 9
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() {
  // Noting 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
  //Mapping the Values between 0 to 255 because we can give output
  //from 0 -255 using the analogwrite funtion
  ledVal = map(val, 0, 1023, 0, 255);
  analogWrite(ledPin, ledVal);
  delay(10);
}
```

6c. Servo Motor Control

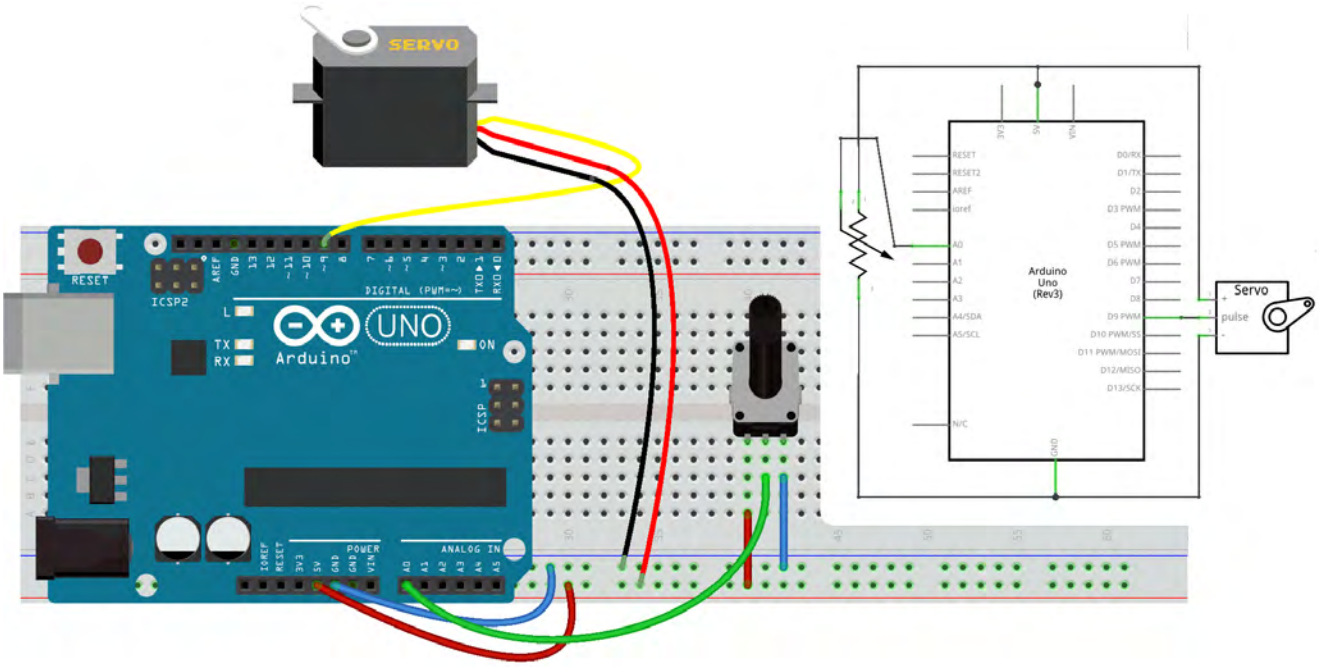
Nu gaan we de LED vervangen door een **Servo Motor**.

Een servo is een motor met een as die naar een opgegeven positie kan draaien. Meestal hebben servo's een bereik van 0 tot 180 graden. Met een Arduino kunnen we een servo naar een specifieke positie sturen. We zullen zien hoe je een servo aansluit en hoe je hem laat draaien naar verschillende posities op basis van de waarde van onze potentiometer.

Circuit

Onze servo heeft een vrouwelijke connector met drie pinnen:

- De donkerste draad (hier bruin) is meestal ground. Verbind deze met GND op de Arduino.
- De voedingsdraad, die volgens de standaard rood is, verbind je met 5V op de Arduino.
- De overblijvende draad op de servo-connector verbind je met digital pin 9 op de Arduino.



Let op: servo's kunnen behoorlijk veel stroom verbruiken. Als je meer dan één of twee servo's wilt aansturen, heb je waarschijnlijk een aparte voeding nodig (dus niet via de +5V-pin van je Arduino).

Code

In this example we will use **a specific servo library** that will make coding a lot easier.

To use a library in a sketch, select it from Sketch > Import Library or just type in the `#include <name_of_library>` command.

```
#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);
  }
```

Zie de Servo referentiepagina voor meer informatie over het gebruik.

De belangrijkste functies die hier worden gebruikt zijn:

- `Servo objectname;` - Hiermee maak je een servo-object aan.
- `objectname.attach(interface)` - Hiermee selecteer je de pin voor de servo. Dat kan in principe met elke digitale IO pin maar vaak worden pin 9 of 10 gebruikt.
- `objectname.write(angle)` - Hiermee stel je de hoek van de servo in (van 0 tot 180 graden).