# Achtergrondinformatie (alle grijze tekst mag je overslaan)

Als HBO ICT-er met afstudeerrichting Technische Informatica leer je software te maken voor intelligente apparaten en complexe computerprogramma’s. In het komende half uur gaan we aan de slag met het programmeren van een microcontroller. Een microcontroller is een hele kleine computer die in je in veel intelligente apparaten aantreft. Voor deze workshop maken we gebruik van een Arduino Uno minicomputer die een microcontroller bevat.

## Intelligente apparaten

In de wereld om ons heen worden apparaten steeds slimmer. Waar je vroeger een radio met de hand moest bedienen, kun je nu je audio systeem met je smartphone vanaf de hele wereld bedienen.

Maar denk ook eens aan printers die nu ook 3D  
producten kunnen maken. Of aan een slimme lepel die  
ook de soep op de lepel houdt als iemand een trillende  
hand heeft. Door deze lepel kunnen Parkinson-  
patiënten zelfstandig blijven eten.

## Microcontrollers

De basis van intelligente apparaten is bijna altijd een *microcontroller*, een computer die volledig op één chip past en daarom makkelijk in apparaten te integreren is. Bovendien gebruiken microcontrollers maar weinig stroom.

Omdat microcontrollers zo klein zijn en zo weinig stroom gebruiken, zijn ze wel stukken minder krachtig dan een ‘gewone’ desktopcomputer of laptop. Een gemiddelde microcontroller bevat zo’n 256 KB opslaggeheugen, 16KB werkgeheugen en heeft een snelheid van 30MHz[[1]](#footnote-1), terwijl een gemiddelde laptop een factor miljoen keer meer geheugen heeft (200 GB opslaggeheugen op de vaste schijf en 4 GB werkgeheugen) en een factor honderd keer sneller is (quad core 2GHz).

Om intelligente apparaten te kunnen maken moet je daarom slim gebruik maken van die beperkte rekenkracht van een microcontroller, en dat leer je bij de afstudeer­richting Technische Informatica. Doordat je leert hoe de bits en bytes van een computer werken, snap je hoe je optimaal gebruik van een computer kunt maken.

## Arduino

Voor deze workshop maken we gebruik van een *Arduino Uno*(revisie 3). De Arduino Uno is een open-source computer, die  
veel door hobbyisten gebruikt wordt. Het kloppende hart van  
de Arduino Uno is de *ATmega328* microcontroller van de firma  
Atmel. Deze microcontroller heeft 32 KB (flash) opslaggeheugen en 2KB werkgeheugen.

De Arduino Uno wordt niet alleen gebruikt voor hobbyprojecten, maar bijvoorbeeld ook voor 3D printers (zoals de RepRap en Ultimaker) en voor het aansturen van drones (zoals de ArduPlane).

# Knipperlicht programmeren

## **Wat gaan we maken?**

Omdat we maar een half uur de tijd hebben, kunnen we  
helaas geen volledig intelligent apparaat maken. Maar om je  
toch een idee te geven van wat een HBO ICT-er met  
afstudeerrichting Technisch Informatica doet, gaan we je wel  
laten proeven aan het programmeren van een micro-  
controller en een leuk spelletje maken.

Je gaat een programma maken waarmee je je reactiesnelheid kunt meten. Van je laptop weet je vast wel hoe snel hij is, maar hoe snel ben je zelf eigenlijk? Leuk om dat te meten en te ontdekken dat je sneller bent dan je vader of je moeder!

## **Inloggen**

Als je deze workshop uitvoert op een computer van de Hogeschool Utrecht, dan kan het nodig zijn om in te loggen op de computer en eventueel in een virtuele machine in te loggen. Hoe dat werkt en welk wachtwoord je mag gebruiken staat op een los A4-tje bij de computer met daarop het wachtwoord voor jouw computer.

## **Arduino Uno aansluiten**

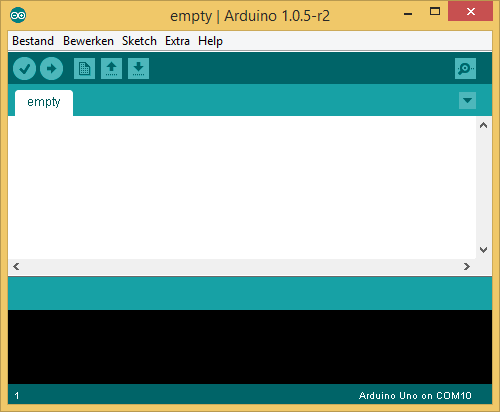
Voordat je de Arduino Uno kunt programmeren, moet je hem aansluiten op de computer. Dit doe je met behulp van een USB kabel. De vierkante stekker gaat in de Arduino, de platte stekker in de computer.

**Let op:** als je werkt onder een virtuele machine, dan moet je in het window van de virtuele machine staan voor je de USB kabel in de computer stopt, anders herkent de virtuele machine jouw Arduino niet.

## **Arduino IDE opstarten**

Computerprogramma’s voor een Arduino heten ook wel sketches. Om een sketch in te voeren en naar de Arduino te zenden maak je gebruik van een ontwikkelomgeving. De Engelse benaming voor ontwikkelomgeving is *IDE* (*integrated development environment*).

Je start de IDE op door te klikken te klikken het  icoontje (op bureaublad). Van het IDE-scherm gaan we tijdens de workshop drie onderdelen gebruiken: de knoppenbalk, het invoerscherm en het uitvoerscherm:



knoppenbalk

invoerscherm

uitvoerscherm

In het invoerscherm voer je de tekst van je sketch (programma) in. Vervolgens gebruik je de knoppen van de knoppenbalk om je programma te controleren en naar je Arduino te zenden. Als er tijdens het controleren iets mis gaat, dan kun je in het uitvoerscherm zien wat er aan de hand is.

De betekenis van de knoppen is als volgt:  
 controleren of jouw sketch (programma) correct is  
 uploaden van jouw sketch naar de Arduino  
 nieuwe sketch aanmaken  
 opslaan van jouw sketch  
 open de seriële monitor om de uitvoer van de Arduino te bekijken

## **Hello world sketch invoeren**

We gaan beginnen met een eenvoudige sketch, die de boodschap “Hello world” naar de computer terugstuurt. Je hoeft de tekst na de // niet in te typen, twee slashes geven aan dat er alleen nog maar commentaar (voor mensen) op deze regel volgt.

Voer onderstaande programma in (in het invoerscherm) en druk daarna op de knop om het controleren ():

void setup() { // (1)

Serial.begin(9600); // (2)

}

void loop() { // (3)

Serial.println("Hello world"); // (4)

delay(10 \* 1000); // (5)

}

Als de titelbalk van je uitvoerscherm groen kleurt en je de boodschap “Compileren voltooid“ ziet, dan is alles goed gegaan en heb je het programma goed ingevoerd. Als je een rode melding in de titelbalk krijgt, dan heb je waarschijnlijk een typefout gemaakt. Controleer je programma en let daarbij goed op de {, }, " en ; tekens.

Een foutmelding in het uitvoerscherm geeft precies aan wat er fout is gegaan:

de naam van het bestand waar de fout in staat (sketch\_nov11a)

sketch\_nov11a**:**7**:** error: expected `;' before 'delay'

het regelnummer in het bestand waar de fout staat

omschrijving van de fout (hier: ‘;’ ontbreekt…)

Als alles goed gegaan is, zie je in het uitvoerscherm hoe groot de machinecode van jouw sketch geworden is:  
Binaire sketch-grootte: x.xxx bytes (van een 32.256-byte maximum)

We gaan nu eerst de sketch testen en na het testen van de sketch leggen we de werking van je sketch stap voor stap uit.

## **Hello world sketch uitvoeren**

Nu de sketch goed hebben ingevoerd, gaan we de sketch compileren en naar de Arduino Uno versturen. We doen dat door de knop uploaden () in te drukken.

Als je nu een foutmelding krijgt (maar het controleren wel goed gegaan is), dan zie je waarschijnlijk de melding: “Seriële poort 'COMx' niet gevonden. Heeft u de juiste poort geselecteerd uit het menu Extra > Seriële poort? aangesloten?” Kijk dan of de USB kabel goed is aangesloten. Als je onder een virtuele machine werkt is het ook van belang om te kijken of het venster van de virtuele machine actief was (je er met de muis instond) toen je de Arduino aansloot. Als dat niet het geval was, haal dan de USB stekker uit de computer, selecteer het venster van de virtuele machine en steek hem er vervolgens weer opnieuw in. Als het daarna nog steeds niet werkt, voer dan samen met een begeleider stap 4 uit van de handleiding *Getting Started with Arduino on Windows* (de handleiding is te vinden op Internet: arduino.cc/en/Guide/Windows).

Als alles goed gaat zie je de boodschap “Upload voltooid” staan.

Start nu de seriële monitor (het uitvoerscherm) op, door op de seriële monitor-knop te drukken (). Als je gekke tekens op het uitvoerscherm/seriële monitor-scherm ziet, controleer dan of de snelheid (rechtsonder in het scherm) op 9.600 baud staat.

Als het goed is zie je nu iedere tien seconden de boodschap: “Hello world” op het scherm verschijnen.

## **Hoe werkt de Hello world sketch?**

We hebben zojuist onderstaande sketch ingevoerd:

void setup() { // (1)

Serial.begin(9600); // (2)

}

void loop() { // (3)

Serial.println("Hello world"); // (4)

delay(10 \* 1000); // (5)

}

### Uitleg:

1. Iedere Arduino sketch bevat één setup() functie en één loop() functie. De setup() functie wordt eenmaal uitgevoerd na het aanzetten om de Arduino-computer in te stellen.
2. Ieder commando van een Arduino sketch eindigt met het ; teken.  
   Met het commando Serial.begin(9600) wordt de snelheid van de (seriële) verbinding tussen de desktopcomputer en de Arduino ingesteld.
3. Nadat de setup() functie is uitgevoerd, wordt zonder onderbreking continu de functie loop() aangeroepen om de sketch uit te voeren op de Arduino. Dit noemen we ook wel de *infinite loop* of *main loop*.
4. Met het commando Serial.println("Hello world") wordt de tekst “Hello world” naar de computer verzonden.
5. Om te voorkomen dat de boodschap heel vaak verstuurd wordt, wordt er daarna met delay(10 \* 1000) steeds 10 seconden (10.000 milliseconden) gewacht voordat de Arduino verder gaat.

## **Hello world sketch opslaan**

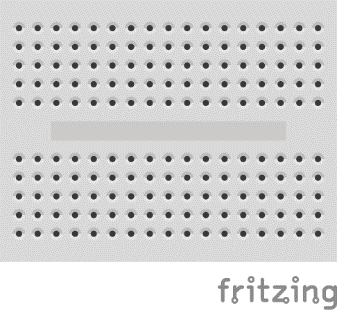
Nu we klaar zijn met het maken van de eerste sketch, kun je deze opslaan met behulp van de opslaan-knop (). Geef de sketch een goede naam en sla hem op het bureaublad. Je kunt nu een nieuwe sketch maken met de nieuw-knop () of door de bestaande sketch een tweede keer op te slaan, maar nu onder een andere naam (je kunt dan verder gaan waar je gebleven was).

Het kan zijn dat je de volgende melding krijgt: “De naam van de sketch moest worden gewijzigd. Sketch-namen mogen alleen ASCII-karakters en cijfers bevatten (en kunnen niet beginnen met een cijfer). De naam moet ook korter dan 64 karakter zijn.“ Deze melding mag je negeren, de IDE heeft de naam van je sketch iets aangepast, omdat hij anders niet correct zou zijn.

## **Hello world sketch aanpassen**

**Vraag:** *Hoe kun je de “Hello world”-sketch zo aanpassen dat de boodschap iedere 5 seconden op het scherm verschijnt?*

## **Aansluiten van een drukknop**



breadboard

5 gaatjes op één rij zijn verbonden

Na onze eerste sketch geschreven te hebben, gaan we met de Arduino een drukknop gebruiken. Om elektronica aan te sluiten maken we gebruik van een *breadboard*. Een breadboard is een plastic blokje met gaatjes, waarvan iedere rij met vijf gaatjes onderling (elektrisch) verbonden zijn. Als je dus twee draadjes op één rij naast elkaar in het breadboard steekt, dan zijn de draadjes met elkaar verbonden.

Het breadboard zit vastgeplakt op een printplaat die verbonden is met de Arduino Uno. De printplaat is bedoelt om schakelingen met de Arduino Uno mee te maken en heet een *prototype shield*. Via de zwarte connectoren kun je een component (zoals een drukknop) aan de Arduino koppelen. We maken vandaag vooral gebruik van de connector aan de rechter kant, waar de digitale (GPIO) pinnen van de Arduino zitten (deze zijn genummerd van D 0 tot D 13, met D0 rechtsondering). Onder het breadboard zit ook toegang tot stroom: linksonder *GND* (GND is de afkorting voor ground, Engels voor *aarde*, oftewel 0v), rechtsonder 5v.

### Aansluiten:

* Plaats de drukknop in het breadboard. Zorg ervoor dat de dichte kant van het plastic kapje van de drukknop naar de groef van het breadboard wijst en één van de openingen in het kapje naar GND wijst (anders is hij een kwartslag verkeerd gedraaid).
* Plaats een *steekbrug* (een elektriciteitsdraadje) op het breadboard zodat het bovenste contact van de drukknop met connector D 4 verbonden is.
* Plaats een steekbrug op het breadboard zodat het onderste contact van de drukknop met connector GND verbonden is.

Nu is de schakeling klaar.

## **Sketch voor het uitlezen van een drukknop**

We gaan een sketch maken die de status van de drukknop uitleest en de status teruggeeft via de seriële monitor.

Om de drukknop uit te lezen, hebben we deze geplaatst tussen de aarde (GND) en connector D 4. Connector D 4 is één van de *GPIO*-pinnen (*general purpose input/ output*) van de Arduino. Met een GPIO-pin kun je of een component aansturen (door er 0v of 5v naar toe te sturen) of component mee uitlezen (door te kijken of er spanning op de GPIO-pin staat). Als de drukknop is ingedrukt, dan wordt de spanning op GPIO-pin 4 gelijk gemaakt aan 0v, anders blijft hij 5v, de standaard spanning voor deze GPIO-pin als hij nergens mee verbonden is.

Type onderstaand programma over in het invoerscherm en druk daarna op de knop om het te controleren ():

void setup() {

Serial.begin(9600);

pinMode(4, INPUT\_PULLUP); // (1)

}

void loop() {

Serial.println(digitalRead(4)); // (2)

delay(2 \* 1000);

}

Zodra we de sketch goed hebben ingevoerd, gaan we de sketch compileren en naar de Arduino Uno versturen. We doen dat door de knop uploaden () in te drukken. Open nu de seriële monitor door op de seriële monitor-knop te drukken (). Kijk wat er gebeurt als je de knop indrukt.

### Voorbeeld uitvoer op seriële monitor

1 knop is niet ingedrukt

1

1

0 knop is ingedrukt

0

0

### Uitleg:

1. Omdat een GPIO-pin zowel als invoerpin als uitvoerpin kan dienen, moet je in de setup() functie dat je de desbetreffende pin als input-pin wil gebruiken. Dat doe je met de opdracht pinMode(4, INPUT\_PULLUP).
2. Om de waarde van GPIO-pin uit te lezen moet je de functie digitalRead(*pin*) aanroepen. Deze wordt in deze sketch meteen getoond op de seriële monitor.

## **Sketch voor het uitlezen van een drukknop, aangepast met constante**

In de sketch, die we tijdens de vorige stap gemaakt hebben, zit nog een tekortkoming: overal in de code hebben we aangegeven met welke GPIO-pin de button verbonden is. Als we later de button op een andere GPIO-pin willen aansluiten (bijvoorbeeld om te voorkomen dat steekbruggen in de knoop raken) hebben we een probleem: we moeten dan de hele code doorzoeken om het pinnummer te vervangen.

Dat kan beter: we kunnen het nummer van de pin (in geval 4) op een plaats in het geheugen wegschrijven. Deze plaats kunnen we een naam geven (bijvoorbeeld buttonPin). Vervolgens gebruiken we alleen nog maar de naam van deze geheugenplaats in plaats van de waarde zelf. Als we later de drukknop aan een andere pin willen hangen, dan hoeven we nog maar op één plaats de waarde aan te passen.

Om te weten hoeveel geheugen we nodig hebben om iets in een geheugenplaats op te slaan, moeten we ook aangeven wat voor soort waarde we willen opslaan. In dit geval willen we een geheel getal in het geheugen opslaan, zo’n getal heet in een sketch een int. Bovendien willen we niet dat de waarde tussentijds wordt aangepast, en dat kunnen we in een sketch aangeven met het woord const.

Deze aanpassing leidt tot de volgende sketch:

const int buttonPin = 4; // (1)

void setup() {

Serial.begin(9600);

pinMode(buttonPin, INPUT\_PULLUP); // (2)

}

void loop() {

Serial.println(digitalRead(buttonPin)); // (3)

delay(2 \* 1000);

}

### Uitleg:

1. Definiëren van geheugenplaats buttonPin, met waarde 4, van het type int, (geheel getal) en waarvan we met const aangeven dat de waarde niet mag worden aangepast.
2. In deze regel geven we aan dat pinMode() niet zomaar pin 4 moet instellen, maar pin buttonPin.
3. In deze regel geven we aan dat digitalRead() niet pin 4 moet uitlezen, maar pin buttonPin.

## **Sketch voor het uitlezen van een drukknop, aangepast met variabele**

In de volgende sketch laten we zien dat we waardes die we uitlezen met digitalRead() niet meteen hoeven te gebruiken, maar ook kunnen opslaan in het geheugen. Dat kan bijvoorbeeld nuttig zijn als we de waarden twee keer zouden willen gebruiken.

Het tussentijds opslaan van waardes zien we in de volgende sketch:

const int buttonPin = 4;

int buttonValue; // (1)

void setup() {

Serial.begin(9600);

pinMode(buttonPin, INPUT\_PULLUP);

}

void loop() {

buttonValue = digitalRead(buttonPin); // (2)

Serial.println(buttonValue); // (3)

delay(2 \* 1000);

}

### Uitleg:

1. Definiëren van aanpasbare geheugenplaats buttonValue, van het type int, (geheel getal). Omdat de geheugenplaats niet const is, mag de inhoud worden aangepast.
2. In deze coderegel geven we aan dat digitalRead() niet de inhoud meteen moet teruggeven aan Serial.println() maar in plaats daarvan de waarde van buttonValue moet instellen.
3. In deze coderegel geven we aan dat Serial.println() niet het resultaat van digitalRead() maar de waarde van buttonValue moet afdrukken.

## **Sketch voor uitlezen indrukknop, aangepast met if statement**

In de voorafgaande sketches zien we 0 af als we de drukknop ingedrukt hebben en 1 als we de drukknop niet ingedrukt hebben. In de onderstaande sketch gaan we een betere melding sturen naar de gebruiker, namelijk “knop ingedrukt“ of “knop niet ingedrukt”.

Om twee verschillende boodschappen te kunnen tonen, moeten we een keuze kunnen maken, namelijk of we Serial.println("knop ingedrukt") uitvoeren of Serial.println("knop niet ingedrukt"). Om een keuze in een sketch te kunnen maken, maken we gebruik van de volgende constructie:  
if (*conditie*) {  
 ...  
}  
else {  
 ...  
}

Als *conditie* de waarde true , of de waarde LOW of een numerieke waarde ongelijk aan 0 (bijvoorbeeld 1) heeft, dan voeren we de code tussen de eerst accolades { ... } uit, anders de code tussen de accolades na else. We mogen het else deel overigens ook weglaten.

Voer de volgende sketch in:

const int buttonPin = 4;

int buttonValue;

void setup() {

Serial.begin(9600);

pinMode(buttonPin, INPUT\_PULLUP);

}

void loop() {

buttonValue = digitalRead(buttonPin);

if (buttonValue) { // (1)

Serial.println("knop niet ingedrukt"); // (2)

}

else { // (3)

Serial.println("knop ingedrukt"); // (4)

}

delay(2 \* 1000);

}

### Uitleg:

1. Het if statement, waarin wordt getest of de conditie buttonValue ongelijk is aan 0.
2. Deze code wordt uitgevoerd als buttonValue ongelijk is aan 0 (de knop is dan niet ingedrukt).
3. Het else deel, dat wordt uitgevoerd als de conditie buttonValue gelijk is aan 0.
4. Deze code wordt uitgevoerd als buttonValue gelijk is aan 0 (de knop is dan wel ingedrukt).

## **Aansluiten van een LED en een weerstand**

In deze stap gaan we een LED (een soort lampje) aansluiten op de Arduino, zodat we een lampje kunnen aansturen. Om de LED naar behoren te laten werken zullen we ook met een weerstand moeten werken.



korte kant => -  
(kathode)

lange kant => +  
(anode)

Een LED is een andere naam voor light-emitting diode. Een LED geeft licht als er spanning op de pootjes van de LED staat. Wat het gebruik van een LED iets lastiger maakt is dat een LED richtingsgevoelig is (net als alle andere diodes). Daarom gaat de LED alleen branden als er een positieve spanning op de lange poot staat van de LED staat (de lange poot is de anode of pluszijde van een LED).

Een andere moeilijkheid van LEDs is dat zij kunnen doorbranden als er teveel spanning op de pootjes van de LED komt te staan. Uit de GPIO-pinnen van de Arduino komt een spanning van 5v, terwijl LEDs doorbranden als er 2v of meer op wordt aangesloten. Om te voorkomen dat we de LED doorbranden, moeten we er daarom een weerstand voor plaatsen.

Het doel van een weerstand is om elektriciteit ‘af te remmen’. Door een weerstand voor of na een component te plaatsen neemt de stroom af. Dit kan nuttig zijn om te voorkomen dat een component anders overbelast wordt. De maten van afremming druk je uit in Ω (Ohm), hoe hoger de weerstand, hoe groter de afremming.

We gebruiken in deze workshop twee soorten weerstanden: weerstanden van 1kΩ (1.000Ω) en weerstanden van 10kΩ (10.000Ω). Ω spreek je uit als Ohm.



1.000Ω/1k Ω

10.000Ω/10k Ω



1.000Ω/1k Ω

10.000Ω/10k Ω

Weerstanden zijn te herkennen aan een kleurcode. Een 1kΩ weerstand heeft als kleurcode bruin/zwart/rood/goud (gele achtergrond) of bruin/zwart/zwart/bruin/bruin (blauwe achtergrond). Een 10kΩ weerstand heeft als kleurcode bruin/zwart/oranje/goud (gele achtergrond) of bruin/zwart/zwart/rood/bruin (blauwe achtergrond).

We gebruiken tijdens deze workshop een 1k Ω weerstand (kleurcode bruin/zwart/rood/goud of bruin/zwart/zwart/bruin/bruin) in serie met de LED om te voorkomen dat de LED doorbrandt.

### Aansluiten:

* Plaats de LED in het breadboard. Zorg ervoor dat de kortste poot van de LED boven is.
* Plaats een steekbrug (een elektriciteitsdraadje) op het breadboard zodat het onderste pootje van de LED met connector D 3 verbonden is.
* Buig een 1k Ω weerstand (kleurcode bruin/zwart/rood/goud of bruin/zwart/zwart/bruin/bruin) in een U-vorm en steek hem in zo het breadboard dat hij contact maakt met het bovenste pootje van de LED (zie schema).
* Plaats een steekbrug op het breadboard zodat het ander contact van de weerstand verbonden wordt met connector GND.

## **Sketch voor aansturen LED**

Nu we de LED en de weerstand aan de Arduino gekoppeld hebben, moeten we een sketch naar de Arduino sturen die ervoor zorg dat de LED gaat branden.

Voer onderstaande sketch in (nadat je de vorige sketch hebt opgeslagen):

const int ledPin = 3; // (1)

void setup() {

pinMode(ledPin, OUTPUT); // (2)

}

void loop() {

digitalWrite(ledPin, HIGH); // (3)

}

### Uitleg:

1. We definiëren een constante ledPin van het type int om een naam te geven aan de GPIO-pin waaraan de LED is verbonden.
2. Als we een GPIO-pin willen gebruiken om een LED aan te sturen, moeten we aangeven dat we de GPIO-pin als output willen gebruiken. Dan doen we met de opdracht pinMode(ledPin, OUTPUT).
3. In het hoofdprogramma zetten we de LED aan, door de waarde van de GPIO-pin op hoog (HIGH) te zetten met het commando digitalWrite(ledPin, HIGH).

Druk na het invoeren van het programma op de knop om het programma te controleren (). Zodras je de fouten eruit hebt gehaald, druk je op de knop uploaden () om het programma naar de Adruino te verzenden.

Als het uploaden goed gaat maar de LED gaat niet branden, dan heb je waarschijnlijk iets niet goed aangesloten. Controleer je schakeling uit voorafgaande stap zorgvuldig. Als het dan nog steeds niet werkt, dan kan een van de begeleiders met een digitale multimeter controleren waar het probleem zit.

Als de sketch goed werkt, mag je (als je dat wilt) de Arduino loskoppelen van de desktopcomputer en aansluiten op een netvoeding (zwarte stekker). Zo kun je zien dat niet stiekem de desktopcomputer alles op de achtergrond aanstuurt.

**Let op:** *zorg ervoor dat niet tegelijkertijd de USB kabel en de netvoeding met de Arduino verbonden zijn. Zo voorkom je dat als er dan iets fout gaat met de netvoeding je de desktopcomputer beschadigt.*

## **Sketch voor aansturen knipperende LED**

In de stap gaan we de LED laten knipperen. Dat doen we door om beurten de LED aan en daarna weer uit te zetten.

Tussen dat aan-en uitzetten moeten we wel wachten met het commando delay(), anders gaat de LED zo snel aan en uit dat je het niet met je ogen kunt zien.

Omdat de main loop toch steeds opnieuw wordt aangeroepen, hoef je de LED maar één keer aan en daarna weer uit te zetten. Het steeds weer aanroepen van de loop zorgt ervoor dat de LED gaat knipperen.

Voer onderstaande sketch in (nadat je de vorige sketch hebt opgeslagen):

const int ledPin = 3;

void setup() {

pinMode(ledPin, OUTPUT);

}

void loop() {

digitalWrite(ledPin, HIGH); // (1)

delay(0.5 \* 1000);

digitalWrite(ledPin, LOW); // (2)

delay(0.5 \* 1000);

}

### Uitleg:

1. We zetten we de LED aan, door de waarde van de GPIO-pin op hoog (HIGH) te zetten met het commando digitalWrite(ledPin, HIGH).
2. We zetten we (na even gewacht te hebben) de LED weer uit, door de waarde van de GPIO-pin op laag (LOW) te zetten met het commando digitalWrite(ledPin, LOW).

## **Sketch voor het aansturen van LED met drukknop**

In deze stap gaan we een LED aansturen door een drukknop in te drukken. Als je de drukknop indrukt, dan gaat de LED branden en als je de drukknop weer loslaat gaat de LED weer uit. Voer onderstaande sketch in:

const int buttonPin = 4;

const int ledPin = 3;

int buttonValue;

void setup() {

pinMode(buttonPin, INPUT\_PULLUP);

pinMode(ledPin, OUTPUT);

}

void loop() {

buttonValue = digitalRead(buttonPin);

if (buttonValue) { // (1)

digitalWrite(ledPin, LOW); // (2)

}

else {

digitalWrite(ledPin, HIGH); // (3)

}

}

### Uitleg:

1. In de geheugenplaats buttonValue staat (nadat hij door digitalRead() is gevuld) of de drukknop is ingedrukt. Waarde HIGH/true betekent dat de knop niet is ingedrukt. Met het if commando testen we of de knop is ingedrukt.
2. Als de knop niet is ingedrukt zetten we de LED uit met commando digitalWrite(ledPin, LOW).
3. Als de knop wel is ingedrukt zetten we de LED aan met commando digitalWrite(ledPin, HIGH).

**Vraag:** *Wat gebeurt er als je de regel "if (buttonValue) {" vervangt door de regel  
"if (! buttonValue) {"?*

**Vraag:** *Wat moet je dan nog meer aanpassen (na het toevoegen van het ! teken) om het programma te laten werken als het voorheen deed?*

**Vraag:** *Hoe zou je door, voorafgaande sketches te combineren, een sketch kunnen maken die het LED licht laat knipperen, als je de knop indrukt?*

## **Sketch voor het meten van je reactiesnelheid**

Bij de vorige opdracht heb je de LED laten branden als je de drukknop indrukt. Dat lijkt heel eenvoudig maar je hebt de aansturing gedaan door in de sketch de drukknop te lezen en daarbij de LED vanuit de Arduino aan te sturen. Bij deze opdracht doe je hetzelfde, maar dan in een andere volgorde. Je zet de LED vanuit je sketch aan en meet hoelang het duurt voordat de drukknop wordt ingedrukt. Het resultaat van de meting stuur je naar het computerscherm. Zo meet je je reactiesnelheid.

Deze sketch bestaat meerdere delen. Je start met het aanzetten van de LED op een (onbekend) moment. Daarna start je de meting die telt in stappen van 1 milliseconde. Je stopt de meting als de drukknop wordt ingedrukt. Vervolgens stuur je het resultaat van de meting naar het scherm van de computer.

Alle stappen heb je in de voorgaande opdrachten al geoefend. Het lastige van deze opdracht is de meting. Daarbij wordt een ‘teller’ bij iedere milliseconde met één stap verhoogd, maar moet je óók de drukknop continu in de gaten houden. De meting moet direct stoppen als de drukknop wordt ingedrukt. Hiervoor gebruik je het ‘while statement’. De wordt uitgevoerd ‘zolang’ de drukknop niet is ingedrukt. Als deze wel wordt ingedrukt stop je de meting en ga je door naar het volgende deel van de sketch: het verzenden van de tijd die je gemeten hebt.

**Vraag:** *Hoe snel denk je dat je bent? De reactietijd wordt op het scherm weergegeven in milliseconden, dus 1000 milliseconden is 1 seconde. Een jonge ervaren gamer heeft een reactiesnelheid die kleiner is dan 0,2 seconde. Op het scherm is dat 200 milliseconde. De belangrijkste vraag is natuurlijk: ben je sneller dan je vader of je moeder?*

En dan nu het programma:

int meting;

void setup() {

Serial.begin(9600);

pinMode(4,INPUT\_PULLUP); //pin 4 is ingang voor de drukknop

pinMode(3,OUTPUT); //pin 3 is uitgang voor de LED

Serial.println("Wacht tot de LED brandt en druk dan snel op de knop!");

}

void loop() {

digitalWrite(3,LOW); // LED uit

delay(9\*1000); // 9 seconden wachten

digitalWrite(3,HIGH); //LED gaat aan, meting start

meting = 0; //meetwaarde resetten

while (digitalRead(4)){ // meting loopt tot knop wordt ingedrukt

delay(1);

meting = meting + 1;

}

Serial.print("uw reactietijd in milliseconden: ");

Serial.println(meting);

## **En verder?**

Als je het leuk vindt kun je nog wat experimenteren met het bovenstaande programma van de reactiemeter. Zoals je kunt zien wordt er in de sketch 9 seconden gewacht voordat de LED wordt aangezet. Dat is een bewuste keuze, deze tijd is lang genoeg om niet meer te weten wanneer de LED aan gaat. Veel mooier is natuurlijk een ‘random’ tijd. Dan weet je niet wanneer de LED gaat branden.

Uit ervaring blijkt dat een kortere tijd een betere score oplevert. Probeer bijvoorbeeld maar eens 3 seconde in te stellen. Bij deze tijd kun je gemakkelijk aftellen en redelijk schatten wanneer de LED aan gaat.

# Antwoorden op vragen

## Hoe kun je de “Hello world”-sketch zo aanpassen dat de boodschap iedere 5 seconden op het scherm verschijnt?

Door de opdrachtregel  
 delay(10 \* 1000);  
te vervangen met de regel  
 delay(5 \* 1000);

## Wat gebeurt er als je de regel "if (buttonValue) {" vervangt door de regel "if (! buttonValue) {"?

Het lampje gaat aan als je de knop loslaat en gaat branden als je hem indrukt. Het uitroepteken (!) draait de waarden ja (true/HIGH) en nee (false/LOW) om.

## Wat moet je nog meer aanpassen (na het toevoegen van het ! teken) om het programma te laten werken als het voorheen deed?

const int buttonPin = 4;

const int ledPin = 3;

int buttonValue;

void setup() {

pinMode(buttonPin, INPUT\_PULLUP);

pinMode(ledPin, OUTPUT);

}

void loop() {

buttonValue = digitalRead(buttonPin);

if (!buttonValue) {

digitalWrite(ledPin, HIGH); // waarden van HIGH en LOW

} // zijn omgedraaid bij

else { // digital write.

digitalWrite(ledPin, LOW);

}

}

## Hoe zou je door, voorafgaande sketches te combineren, een sketch kunnen maken die het LED licht laat knipperen, als je de knop indrukt?

const int buttonPin = 4;

const int ledPin = 3;

int buttonValue;

void setup() {

pinMode(buttonPin, INPUT\_PULLUP);

pinMode(ledPin, OUTPUT);

}

void loop() {

buttonValue = digitalRead(buttonPin);

if (!buttonValue) {

digitalWrite(ledPin, HIGH);

delay(0.5 \* 1000);

digitalWrite(ledPin, LOW);

delay(0.5 \* 1000);

}

else {

digitalWrite(ledPin, LOW);

}

}

# Thuis verder aan de slag met een Arduino?

Als je het werken met een Arduino leuk hebt gevonden en je zou thuis verder willen gaan met het programmeren van een microcontroller, dan zou je kunnen overwegen om zelf een Arduino Uno aan te schaffen.

Arduino Uno’s zijn onder andere te koop bij online elektronicaleveranciers (zoals Conrad, www.conrad.nl, en Hackerstore, www.hackerstore.nl, of DX, www.dx.com) of een gespecialiseerde elektronicawinkel (zoals Radiocentrum Utrecht in Utrecht). Je kunt een losse Arduino bestellen, maar je bent misschien voordeliger uit als je meteen een starterkit koopt, waarin ook allerlei elektronische componenten zitten die je op de Arduino kunt aansluiten.

# Meer informatie over de opleiding HBO ICT of de afstudeerrichting Technische Informatica?

Als je nog meer wilt weten over de opleiding HBO ICT of het afstudeerrichting Technische Informatica, dan kun je terecht op onze studiekeuze website, http://www.studiekeuze.hu.nl/. Ook kunnen we je van harte aanbevelen om, na deze open dag, je aan te melden voor een meeloop-dag. Je kunt dan een dag ervaren hoe het echt is om hier te studeren. Aanmelden voor een meeloop-dag kan ook via onze website.

# Colofon

## Gebruikte software

Voor het ontwikkelen van deze workshop is gebruik gemaakt van Arduino IDE 1.0.6 en Fritzing 0.9.0b software en van Arduino Uno rev 3 hardware.

## Bronvermelding

Foto op kaft: [Snootlab, *Arduino Uno*. FlickR, 17 augustus 2011, cc-by 2.0](https://www.flickr.com/photos/66478851@N06/6052455554/).

1. Wikipedia (2014), Atmel AVR. Wikipedia, 24 oktober 2014. url: http://en.wikipedia.org/wiki/Atmel\_AVR [bekeken op 1 november 2014] [↑](#footnote-ref-1)