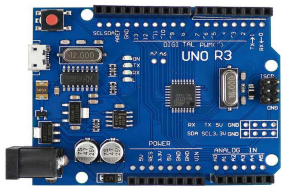
Opdrachten practicum Embedded Systems (met Arduino[1])

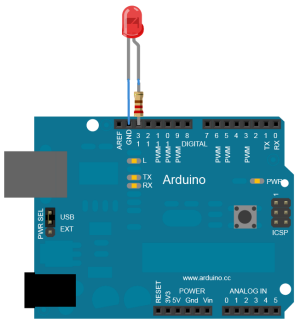
1. Alle cursusmateriaal is te vinden op Black Board
- SOI_HICT_18_57: HBO-ICT & ITSM - PROPEDEUSE => module IOT
- TECH_E_15_447: Domotica 2018-2019 => Information => Embedded systems

2. Ga naar de Arduino homepage (www.arduino.cc), naar software -> downloads en download en installeer de (nieuwste versie van de) Arduino-software. Bekijk, voor algemene informatie, resources -> getting started en resources -> tutorials, vooral build-in examples. Om snel met de Arduino-omgeving vertrouwd te raken beginnen we met het voorbeeld “Blink”, een knipperende LED. Dit is de “Hello World” voor embedded systems.



3. Voor zowel het practicum als het project gebruik je je eigen (zelf aangeschafte) Arduino-bordje, met USB-kabeltje. Start de Arduino-IDE (het groene schermpje) en verbind je Arduino-bordje met je laptop (beide USB). Lees je eerst in (via www.arduino.cc en/of YouTube), sluit dan de Arduino pas aan. Voor je kunt beginnen met programmeren moet je twee instellingen invoeren, die nadien meestal wel worden onthouden. Dat zijn, onder het menu-item Tools, Board en Port. Het board-type is, als je de adviezen hebt opgevolgd, waarschijnlijk Uno. Bij Port kies je de USB-port waaraan je Arduino-bordje hangt. Dat is meestal iets met Arduino of Serial in de naam.
   Let op:
   a) Je koppelt elektronica aan je eigen laptop. Wees voorzichtig met kortsluitingen, voedingen, aansluitingen met + en -, etc. In tegenstelling tot softwarecomponenten, kunnen elektronica-componenten *echt* stuk gaan en, vaak erger nog, andere apparaten beschadigen. Vraag in geval van twijfel de docent. Zorg voor de juiste drivers. Dit gaat overigens meestal “vanzelf” goed.
   b) Lever na afloop van de les geleende hardware, zoals ledmodules, sensoren en kabeltjes weer in.

4. **Blink**
   Voer, na aansluiten en configureren van de hardware, het Blink-voorbeeld uit. Dat staat onder File -> Examples -> Basics -> Blink. Na compilatie en uploading gaat er een klein (geel) LEDje knipperen. Dat LEDje bevindt zich op het Arduino-board (pin13). Er is voor Arduino, en zeker voor de “blinking-LED-opdracht”, veel ondersteuning te vinden op internet. Kijk bij “Language Reference”, “Arduino Booklet”, YouTube-filmpjes, etc. Probeer bij de uitwerking van dit voorbeeld voortdurend te begrijpen wat je doet en hoe en waarin een embedded-programma verschilt van een “normaal” programma.



Het **doel** van de voorgaande 4 opdrachten is dat je de essentie van embedded systemen begrijpt, in het bijzonder die van de Arduino-omgeving. Controle:
- Hoe wordt het Arduino-bordje aangesloten op een laptop? Pas op met externe voeding in combinatie met USB-voeding. Werkt de driver? Zie je de COM-poort?
- Hoe wordt een sketch geladen? Hoe laad je een voorbeeld-sketch uit het Sketchbook, Examples of Recent?
- Wat is de functie van `loop()` in het programma? En van `setup()` ?

[1] Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments.

- Hoe worden de I/O-pinnen gedeclareerd, geconfigureerd en geïnitialiseerd? Wat doen de functies `pinMode()` en `digitalWrite()`.
- Hoe wordt een programma vertaald/gecompileerd? Waar staat de code? Hoe groot is de code?
- Hoe wordt een programma ge-upload? Hoe is dat te zien? Voel je je vertrouwd in de IDE?

Als je bovenstaande vragen (positief) kunt beantwoorden, kun je vol zelfvertrouwen doorgaan naar de volgende opgaven. Nu begint het echt werk.

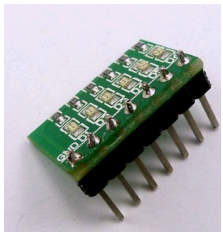
Maak de volgende opgaven uit dit document. Je mag dit in teams van twee studenten doen, maar zorg, wat je taak binnen dit team ook is, dat je op de hoogte bent van de details en deze kunt toelichten bij het aftekenen. Beantwoord ook de kennisvraagjes die bij de diverse opdrachten gesteld worden. Benoemd en archiveer je uitwerkingen zodanig dat je ze terug kunt vinden, zowel voor hergebruik als bij aftekening. Suggestie: opdracht1, opdracht2, etc.

Houd het volgende werk- en aftekenschema aan.

Werken aan in week	**Opdrachten**	**Uiterlijk laten aftekenen (lokaal: zie rooster)**
1 en 2	1 .. 9	Eind week 2, opdrachten 7, 8 en 9
2 en 3	10 .. 16	Eind week 3, opdrachten 12, 15 en 16
3 en 4	17	Eind week 4, opdracht 17
4 en 5	18 .. 19	Eind week 5, opdracht 18 en 19

Opmerking
In een embedded omgeving is het soms lastig om vast te stellen of je programma doet wat je verwacht dat het zou moeten doen. Je hebt immers niet de beschikking over een operating system en allerlei functies zoals debugging of een schermpje waarop je bijvoorbeeld de toestand van een variabele kunt tonen. Bij opgave 9 gaan we, ook in dit verband, kennis maken met de serial monitor. Los daarvan is het gebruik van LED's, om (real time) informatie te zichtbaar te maken, "common use" in een embedded omgeving. Het Arduino-bordje is om die reden ook al uitgerust met een on-board LEDje aan pin 13.

Aanbeveling
Schaf een LED-module aan (de 6 bit multicolor LED module) die eenvoudig gekoppeld kan worden aan de bits 8..13 van PORT B van de Arduino. Daarmee kun je op eenvoudige wijze (6 bits) informatie zichtbaar maken. De nevenstaande module koop je voor minder dan €2. Tijdens de lessen wordt meer informatie verstrekt. Tijdens de practica kun je een dergelijke module lenen, althans zolang de voorraad strekt, maar het is handig zelf over één over aan te beschikken. Even bestellen dus:



https://nl.aliexpress.com/item/2pcs-DC-3-3V-5V-12V-6-Digitals-Blue-LED-Module-6-bits-indicator-Diy-kit/32274439551.html

5. **Timing, frequentie en pulsbreedte**
   De basis voor de volgende opdracht is het voorbeeld van de "Blinking LED".
   a) Sluit een (rode) LED aan tussen pin 13 en de naastliggende GND (Ground). Let op: het langste pootje van de LED is de + (= anode), de kortste de - (=kathode). De + moet aan 13, de - aan GND. Stel vast dat, wanneer het programma loopt, de beide LEDs synchroon knipperen. Wat is de knipperfrequentie?
   Je mag, i.p.v. een losse LED, ook de LED-module gebruiken.

Sluit de (rode) LED nu aan tussen pen 12 en de GND, uiteraard de + aan pin 12. De gele (on-board) LED laat je uiteraard zitten.

b) Modificeer het programma zodanig dat de rode en (on-board) gele LED synchroon knipperen. Bestudeer de functies `pinMode()` en `digitalWrite()`. Noem de aansluitingen voor de LED's ledPin12 en ledPin13, uiteraard naar het fysieke pinnummer op de connector.

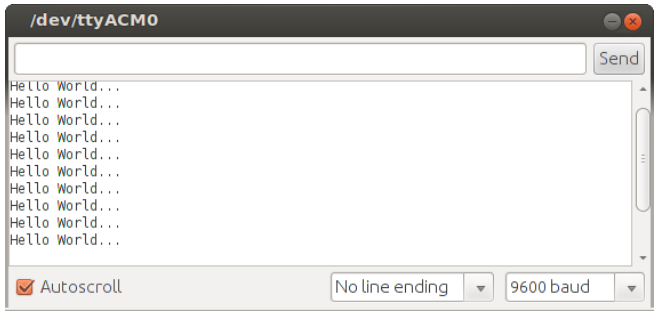
c) Modificeer het programma van de vorige opdracht zodanig dat de rode en de gele LED afwisselend knipperen, dus rood aan/geel uit en rood uit/geel aan. Laat de knipperfrequentie, net als de aan- en uittijden ongemoeid.

d) Modificeer het programma van de vorige opdracht zodanig dat de frequentie van de knipperende LED's 2 Hz wordt, waarbij de rode LED een duty cycle van 25 procent heeft. Dat wil zeggen dat de rode LED 25 procent van de tijd brandt, de gele 75 %. Dit is tevens ter voorbereiding op het begrip "PWM", in opdracht 10.

De Arduino programmeeromgeving heeft een ingebouwde "Serial Monitor". Hiermee kun je informatie vanuit het programma dat op de Arduino loopt zichtbaar maken op je laptop (`Serial.print`). Een erg handige functie, in het bijzonder voor debugging. Hoewel de communicatie tussen laptop en de Arduino verloopt via een USB-kabel, is het onderliggende protocol dat van een seriële poort, soms COM-poort genoemd.
Merk terloops nog even op dat je het gewenste gedrag van de Arduino vastlegt in de chip en dat dat een nogal "kaal gebeuren" is. Je hebt niet de beschikking over Windows (ook al draait de IDE daar wellicht wel onder) en dat er aan de Arduino geen harde schijven en grafisch displays hangen. Je hebt niet de beschikking over een OS, maar onderhand wel over een indrukwekkende hoeveel bibliotheekfuncties en componenten voor heel veel devices en shields. Los van deze constatering lijkt embedded programmering dus wel erg op "gewoon" programmeren. Een student Software Engineering moet beide kunnen. De taal waarin je de Arduino progammeert is "Processing". Het is afgeleid van en lijkt sterk op de taal C en die was er al ruim voor C# ;)
Weet van het bestaan van de Arduino Language Reference en natuurlijk van stackoverflow.

6. Het object "Serial"

De Arduino software beschikt (gelukkig al) over veel bestaande onderdelen, doorgaans componenten genoemd (soms ook library-functies, classes of objecten). Een component bevat ingebouwde functies, vaak methodes genoemd. "Serial" is een Arduino-component waarmee tussen het Arduino-bord en de laptop gecommuniceerd kan worden. Zoals gezegd: vooral handig voor het tonen van (debug)informatie. Om deze component te gebruiken, moet het "ding" geïnitialiseerd worden. Onderzoek hoe je de seriële poort initialiseert. Doe dit, in de functie `setup` (eenmalig dus), op 9600 baud. Onderzoek wat dit betekent. Print vervolgens, m.b.v. `Serial.print`, de mededeling "Hello World…:" of "Lees de Arduino Language Reference" (ook eenmalig, niet dat lezen natuurlijk). Als je de tekst vaker voorbij wilt zien komen, zet deze dan in loop-functie. Probeer maar. 9600 baud is van oudsher een default waarde. Tegenwoordig echter, met veel snellere CPU's, is 115200 baud ook een goeie keuze. Let op dat deze waarde in de programmacode en serial monitor overeen moet komen.



Vraagje: hoeveel tekens kan de Arduino per seconde naar de laptop/serial monitor sturen bij een baud rate van 9600 bps? Om die vraag te kunnen beantwoorden moet je eerst achterhalen met hoeveel bits één teken (bijvoorbeeld 'A', 'a' of '1') eigenlijk gecodeerd wordt? Onderzoek in dat verband ook wat de ASCII-tabel is en welke informatie je daaruit kunt halen. Dit kom terug bij vraag 9. Nu eerst terug naar de basis met o.a. wiskunde en Pythagoras.

7. **Rekenen met de Arduino**
Om te ervaren dat een embedded systeem een computer (CPU) is, weliswaar voor specifieke toepassingen, is het nuttig de embedded controller enkele zaken uit te laten rekenen op een manier zoals je dan met een normale laptop of desktop ook zou kunnen doen.

a) Print de waarde van de oppervlakte van een cirkel (pi * r^2, met r = 4).
Let op de types van de te gebruiken variabelen (`int`, `float`). pi is immers een getal met een fractie (een float) en pi vermenigvuldigd met een getal blijft een float. Kies de types van je variabelen niet groter dan nodig. In een embedded omgeving is het doorgaans woekeren met de recourses. Schrijf je programmafragment in `setup()`. Het heeft weinig zin de oppervlakte in `loop()` eindeloos te blijven doen.

b) Wat wel zin heeft in `loop()`, is het bepalen van de oppervlakte op basis van een variabele r. Schrijf een programmafragment waarmee de oppervlakte wordt berekend en getoond via de seriële monitor voor 1 <= r <= 10. Je kunt gebruikmaken van het herhalende karakter van `loop()`, maar gebruik (daarnaast) ook een traditionele for-loop.

Om duidelijk te maken dat een embedded controller geen onderscheid maakt tussen de personen die het apparaat programmeren en de discipline van waaruit de problemen komen, volgt hier een toepassing vanuit de gaming/graphics/lineaire algebra.

c) Als P(x, y, z) een punt is met de coördinaten x, y, z, dan ligt P op het oppervlak van een bol met middelpunt M = (x_M, y_M, z_M) en straal r als geldt: $r^2 = (x_P - x_M)^2 + (y_P - y_M)^2 + (z_P - z_M)^2$
Schrijf een programmafragment waarmee r wordt bepaald (en getoond) op basis van de volgende punten: M(3, 4, 2) en P(6, 8, 4). (hint: standaard functies: `sq()` en `sqrt()`).
Merk ook op dat we hier te maken hebben met een oude bekende: Pythagoras.

8. **PORTB**
Sluit de LED-module aan op de Arduino. Let weer op de polariteit. We hebben geleerd dat met de functie `digitalWrite()` één enkel bit/pin kan worden aangestuurd. Wat is overigens het verschil tussen een bit en een pin?

a) Schrijf een programma waarmee alle 6 LED's, gekoppeld aan de pinnen 8 t/m 13, eerst afwisselend en daarna tegelijkertijd aan en uit worden gezet. Doe dit met een frequentie van 1 Hz. Opmerkingen:
- vergeet niet alle afzonderlijke pinnen te declareren en te initialiseren (op output).
- het is fysisch onmogelijk om, met gebruik van `digitalWrite`, 6 bits tegelijkertijd aan- of uit te zetten. Het mag dan heel snel gaan, exact is het niet. Dat kan wel met PORTB.
Via de constructie `PORTB` (te gebruiken in een expressie) kunnen meerdere bits ineens aangestuurd worden. Onderzoek het gebruik van het statement `PORTB = value`; waarbij

`value` een integer variabele is.

b) Modificeer het programma zodanig dat het met `PORTB` werkt i.p.v. met `digitalWrite()`.

Opmerking:
Overweeg naast `PORTB` ook `DDRB` te gebruiken. Met `DDRB` kun je meerder pennen tegelijkertijd configureren. Zoek uit wat dat inhoudt, waar DDRB voor staat en hoe dat moet. Als je, bijvoorbeeld, alle LED's, inclusief led13, als output wilt configureren en daarna de overeenkomstige pinnen (en LED's) wilt aan- en uitzetten, dan kan dat op de volgende manier:

```
DDRB  = 0x3F;        // bit 8..13 op output; 14, 15 op input
PORTB = 0x3F;        // 6 LED's aan
PORTB = 0x0;         // 6 LED's uit
```

De waarde `0x3F` geeft de (ingebouwde) variabelen DDBR, resp. PORTB de waarde 63. Het getal 63 is het decimale equivalent van het hexadecimale getal `0x3F`. Merk op dat in het getal 0x3F veel beter de afzonderlijke bits te herkennen zijn dan in 63. Meer over getallen en hun representaties in de volgende opgave. Je had ook mogen schrijven:

```
PORTB = 63;          // decimaal, grondtal 10
```
of in een binaire notatie:
```
PORTB = B00111111;   // binair, grondtal 2
```

Tot slot van deze opgave nog even een detailvraagje over de timing. Waarom zie je, tenzij je bionische ogen hebt, bij de uitvoer van onderstaande statements de LED's niet aangaan?

```
PORTB = 0x0;         // 6 LED's uit
PORTB = 0x3F;        // 6 LED's aan
PORTB = 0x0;         // 6 LED's uit
```

9. **Datarepresentatie en strings**
N.a.v. opdracht 9 heb je waarschijnlijk ontdekt dat de ASCII-tabel informatie biedt over de wijze waarop individuele tekens (Engels: characters) worden gecodeerd. Elk teken heeft zijn eigen unieke code. Zo kun op eenvoudige wijze herleiden dat geldt:

teken	Decimale waarde	Binaire waarde	Hexadecimale waarde
'A'	65	01000001	$41 of 0x41
'a'	97	01100001	$61 of 0x61
'1'	49	00110001	$31 of 0x31
Return	13	00001101	$0D of 0x0D

Merk op dat getallen en dus ook tekens op verschillende manier gerepresenteerd kunnen worden. In het decimale stelsel is het grondtal 10, in het binaire stelsel 2 en in het hexadecimale stelsel 16. De hex. representatie is erg handig, vooral in een embedded omgeving, om lange binaire patronen compact te noteren. 1 hex. digit komt immers overeenkomt met 4 binaire digits ($2^4 = 16$) en dat de getallen 10 t/m 15 daarbij genoteerd worden met A t/m F. Het getal $CA of $(CA)_{16}$ heeft niets met Clements & August Brenninkmeijer te maken, maar duidt op een hexadecimaal getal en komt overeen met $(202)_{10}$, immers C * 16 + 10 = 202.

Deze hex-getallen zijn ook bekend van kleurcoderingen bij plaatjes op computerschermen. Meer informatie tijdens de lessen. We gaan nu eerst praktisch oefenen met strings, characters en hun representatie. We beperken ons daarbij even tot dynamische strings (eigenlijk objecten), omdat dat goed aansluit bij onze C#-ervaringen. Er valt nog veel meer over strings te zeggen, over C-strings (char[]), over pointers, string-terminators en ingewikkelde string functies, maar voorlopig beperken we ons even tot het type `String`.

a) Declareer in de functie `setup()` twee variabelen van het type String.

```
String s1 = "0123456789"; // a constant String
String c1 = 'a';          // a constant String
```

Merk op dat dit constante strings zijn, strings dus met een vaste waarde. s1 is overigens wel aan te passen, maar dat is niet aan te bevelen.

Declareer ook een **object** van het type string:
```
String s2 = String("bcdefghi"); // a String object
```

Merk op dat het keyword `String` aan de rechterkant van het toekenningsstatement de functie heeft van constructor en dat deze string aangepast kan worden, wat ook gaat gebeuren bij de volgende opdracht.

Je kunt een contante string ook converteren naar een object.
```
String c2 = String(c1);          // a String object
```

Druk de vier 4 strings, `s1`, `c1`, `s2` en `c2`, af met `Serial.println()`. Merk op dat de compiler klaagt bij de conversie van het character `'a'` naar een string ("conversion from 'char' to 'String' is ambiguous"). Dat is dus ambigu, wat een moeilijk woord is voor dubbelzinnig is. Herstel de fout en maak van `'a'` een string. Wat wordt geprint?

b) Onderzoek hoe je de string `s2` kunt aanpassen tot een string met de waarde `"abcdefghij"`. Doe dit door bij de string `s2` (aan de voorkant) `c2` "op te tellen" en aan de achterkant de constante string `"j"`. De werking van de operator + is voor variabelen van het type string dus kennelijk uitgebreid met "aan elkaar knopen" of met een moeilijk woord: concatenatie. Druk deze aangepaste string weer af en controleer het resultaat.

c) De elementen van een (variabele van het type) `String` zijn characters. Een (variabele van het type) `String` bestaat uit 0 of meer characters. Een string kan dus ook leeg zijn of leeg worden gemaakt met s = ""; Een (variabele van het type) `String` is een samengesteld type in de vorm van een array van characters. De afzonderlijke elementen kunnen benoemd worden met `s1[i]` (met `0 <= i <= s1.length()-1`) of `c1[i]`, etc. Hieruit valt af te leiden dat het eerste teken van een string op array-positie 0 staat, tenzij de string leeg is natuurlijk, en dat een (variabele van het type) `String` een methode heeft waarmee haar lengte kan worden achterhaald en dat het laatste element van de string op array-positie `s1.length()-1` staat. Het eerste element is dus `s1[0]`, het laatste op `s1[s1.length()-1]`.

Print, steeds op een nieuwe regel, het eerste element van `s1`, het laatste element van `s1` en print daarna, ook steeds op een nieuwe regel, alle afzonderlijke elementen van `s1`. Herhaal dit voor `s2`.

Als het bereik beperkt is (<255) wordt vanwege efficiëntieredenen voor de loop-variable vaak het type `byte` gebruikt i.p.v. `int`.
Pas op met de functie `sizeof()`, waarmee de hoeveelheid geheugenruimte van een variabele kan worden opgevraagd, in combinatie met een (dynamische) string. Gebruik de methode `length()` van het string-object.

d) Print van de afzonderlijke elementen van een string (s1 of s2) de ASCII-waarden. Dat zijn dus niet de (leesbare) tekens (a, b, c,…), maar de overeenkomstige ASCII-waarden (97, 98, 99,…). Dit kun je doen met de functies `int()`. Je kunt deze waarde uiteraard toekennen aan variabele van het type int. Dus na `int h = int(s1[0])` heeft h de waarde `48`. Merk op dat deze waarde decimaal is. Uiteraard geldt: $(48)_{10} = (30)_{16} = (00110000)_2$.
Een andere manier om de decimale waarde van een character te *tonen* is rechtstreeks via een argument van de functie print of `println`, nl.:
`Serial.println(s1[1], DEC).`
Was is het (subtiele) verschil met datgene wat je hierboven doet?
Op overeenkomstige wijze kun je ook de hexadecimale waarde (keyword HEX) of binaire waarde (keyword BIN) tonen.
Druk, op één regel, elk string-element af, gescheiden door spaties, in de vorm van het (normale) teken, gevolgd door de decimale waarde, de hex waarde en de binaire waarde. Dus, bijvoorbeeld:
`a 97 61 1100001`

Let op: met de functies `print` en `printnl` kun je geen variabelen van verschillende datatypes afdrukken.
In de 7-bits binaire representatie is het meest rechtse bit, bit 6, het most significante bit en het meest linkse bit, bit 0, het least significante bit.

e) Druk, tot slot van deze oriëntatie op data representatie en strings, alle characters van een string af op de 6 LED's. Beperk je daarbij tot de bits b0 t/m b5 van een teken, omdat aan we op de LED's maar ruimte hebben voort 6 bits. We laten b6 en b7 dus even schieten. Uiteraard bouw je een delay in voor een prettige 'leesbaarheid'. We zien dus de binaire representatie van de afzonderlijke characters waaruit de string is samengesteld op de LED's verschijnen. Voorbeelden:

teken	dec	hex	bin (7 bits)	p13	p12	p11	p10	p9	p8
				bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
'A'	65	0x41	1 000001	0	0	0	0	0	1
'a'	97	0x61	1 100001	1	0	0	0	0	1
'f'	102	0x66	1 100110	1	0	0	1	1	0
'4'	51	0x33	0 110011	1	1	0	0	1	1

10. **analogWrite**
Sluit de rode LED nu aan op pin 11. D.w.z. de + aan pin 11 en de – aan GND. Je mag ook hier weer de LED-module gebruiken.
Schrijf vervolgens een programma waarmee de rode LED in 5 seconden in helderheid toeneemt van "uit" tot maximale intensiteit. Doe dit in 10 stappen van een halve seconde. De maximale helderheid is (theoretisch) 255. Tip: Beperk dit, voor de eenvoud van de uitwerking, tot 250.
Onderzoek hierbij het gebruik van de functie `analogWrite()`. Enkele vragen:

- Benoem (en declareer) pin 11 in je programma als pwmPin11.
- Waarom volstaat pin 12 voor deze opdracht niet en moet de LED aan pin 11 (of 10, 9, 6, 5, of 3)?
- Wat is de betekenis van de afkorting PWM en wat is de functie ervan? E.e.a. werd al ingeleid in opdracht 8. Formuleer het in je eigen woorden.
- Print tijdens het proces steeds de actuele helderheid (als getal tussen 0 en 250).

Bij de volgende opdrachten gaan we de analoge ingang(en) van de Arduino gebruiken. Daar kun je analoge sensoren op aansluiten. Sensoren zetten een natuurkundige grootheid om naar (meestal) een elektrisch signaal. De Arduino zet op haar beurt, onder besturing van jouw programmmacode, dit elektrische signaal om naar discrete getallen, waar je programma vervolgens iets mee kan doen. Bijvoorbeeld: T (temperatuur in graden Celsius) -> U (spanning in Volts) -> G_1 .. G_N (digitale getallen). Bij deze omzetting zijn de volgende zaken van belang: de sampletijd (tijd tussen twee samples of metingen), het aantal samples (= N) en de resolutie van een sample (aantal bits per sample). Vergelijk dit met bijvoorbeeld de audio-CD; sampletijd 44100 Hz, 16 bits per sample en het aantal samples hangt af van lengte van een song of track.
De resolutie van de AD-converter in de ATMEGA is 10 bits. Dat betekent dat het bereik van de gesampelde input ligt tussen de discrete getallen 0 en 1023.

Bij het practicum wordt een krachtsensor beschikbaar gesteld, maar een eigen sensor (bijvoorbeeld uit je staterskit) mag natuurlijk ook worden gebruikt. De potentiometer (draai- of hoeksensor) leent zich hiervoor het beste, maar een lichtsensor is ook prima. Vergewis je van een goeie aansluiting.
Mocht jouw sensor, bijvoorbeeld door gebruik in een elektronisch schema, een ander bereik hebben, bijvoorbeeld van 400 – 800, weet dan dat je dat op eenvoudige wijze kunt “oprekken” met de functie `map`.

```
valnew = map(val, 400, 800, 0, 1023);
```

11. **analogRead**
    Met de functie `analogRead()` kan informatie van een analoge bron worden ingelezen. Onderzoek de werking van deze functie. Welk argument (= parameter) moet aan de functie worden meegegeven? Wat levert de functie op als resultaat? Van welk type is de functie? Wat is het bereik van het getal dat wordt opgeleverd? Wat is de resolutie (in V/bit), bij gebruik van een 5-Volts sensor.
    Schrijf een programma dat periodiek, met een frequentie van 10 Hz, een waarde van de analoge ingang kanaal 0 leest en print dit resultaat via de Serial Monitor? Hoe verklaar je het resultaat (in het geval er dus nog geen sensor is aangesloten)?
    Maak gebruik van de herhalende werking van de functie `loop()`.

12. **Sensoren (bijv. de krachtsensor)**
    Sluit een analoge (kracht)sensor aan op analoog kanaal 0 van de Arduino. Let op: het rode draadje aan de +5V, de zwarte aan de GND en de witte (het meetsignaal) aan “Analog Channel 0”. Controleer dit met de naamgeving op het Arduino-bordje.
    a) Herhaal de vorige opgave maar nu met de sensor aangesloten. De frequentie blijft 10 HZ. Varieer de sensorwaarde door de grootheid te beïnvloeden (kracht, hoek, licht).
    b) Modificeer het programma zodanig dat je gebruik maakt van de functie `sample()` die een gestandaardiseerde waarde retourneert die ligt tussen 0 en 99.

```
// sample een analoge waarden van kanaal chan
// het functieresultaat is een waarde tussen 0 en 99
```

```
int sample(int chan)
```

Hint: maak gebruik van de functie `map()`.

c) Modificeer het programma zodanig dat de 6 LED's (tijdelijk, kortstondig) gaan knipperen wanneer een maximumwaarde is bereikt. Dat mag 99 of 1023 zijn, maar ook een waarde naar keuze.

d) Modificeer het programma zodanig dat je voor deze alarmeringsfunctie gebruik maakt van de functie `alarm()`.

```
// knipper alle 6 LED nblk keer achter elkaar
// in een "prettige" knipperfrequentie
void alarm(int nblk)
```

Na de alarmering kom je weer terug in de meetlus. Zolang het alarmcriterium van kracht blijft, blijft het alarm zich natuurlijk herhalen.

13. **Sensoren en meetfrequentie**

a) Modificeer het programma van de vorige opgave zodanig dat je de alarmering even uitzet en dat de 6 LED's van PORTB gaan knipperen met een frequentie die wordt bepaald door de sensorwaarde. De sensorwaarde bepaalt daarmee de meetfrequentie (=sample rate). Je mag daarbij gebruik maken van de hierboven ontworpen functie `sample()`. Het effect is dat naarmate de sensor sterker wordt ingedrukt (gedraaid, verlicht, etc.) de LED's langzamer knipperen. Kies frequenties die voor het oog waarneembaar en "prettig" zijn en niet te groot omdat je anders lang moet wachten. Let op: alle LED's gaan nog steeds gezamenlijk aan en uit.

b) Modificeer daarna het programma zodanig dat de beïnvloeding van de knipperfrequentie door sensor net andersom werk. Dus een sterkere kracht (hoek, licht, etc.) geeft een hogere frequentie. Hint: complement.

c) Zet de alarm-functie weer aan, zodat bij overschrijding van een maximumwaarde de LED's gaan knipperen met een vaste frequentie (die niet wordt bepaald door de sensorwaarde).

14. **Datarepresentatie, bits en byte's**

Schrijf een programma, of modificeer bovenstaande, waarmee de (gedigitaliseerde) input van de sensor als getal wordt afgebeeld op de 6 LED's. Kies een "prettige" meetfrequentie die niet meer afhankelijk is van de sensorwaarde.
Feitelijk gaat het hier dan om een binaire afbeelding. Let op: met de zes LED's kan een getal tussen 0 (binair: 000000) en 63 (binair: 111111) worden afgebeeld. De input van de A/D-convertor levert echter getallen op waarvan de resolutie groter is. Wat zijn het bereik en de resolutie van de A/D-convertor in de Arduino ook al weer? Hoe los je het probleem op waarbij informatie van méér dan 6 bits (per getal) moet worden afgebeeld op exact 6 bits?
Overweeg de functie `sample()` uit de vorige opgaven hiervoor te gebruiken.
Tot slot van deze opgave: oefen het binair tellen van 0 tot 15 en vergewis je ervan dat je van de getallen 0 t/m 15 de hexadecimale en binaire representatie kent.

15. **De VU-meter**

Vanuit de "audio-wereld" is het concept van de VU-meter bekend: Een zacht signaal geeft een kleine uitslag, een luid signaal een grote.

a) Ontwerp en schrijf een programma waarmee een (soort van) kracht VU-meter wordt gerealiseerd. Bij een kleine kracht slaan er weinig LED's uit, bij een grote kracht veel. Bij geen kracht zijn alle LED uit, bij maximale kracht 6 LED's aan. Hier dient het begrip maximum te worden geïnterpreteerd als een kracht waarbij je een afdruk van de sensor in je duim mag zien maar zonder bloed. Kortom: jullie hoeven ons er niet van te overtuigen dat je, met het bloed in je handen, de sensor naar de maximale waarde van 1023 kunt

drukken. Het is geen kermisattractie. Trouwens, waar komt die waarde 1023 ook al weer vandaan?

Zorg dat je software zodanig werkt dat de uitvoer "soepel" reageert op de inputveranderingen. Dit heeft o.a. te maken met de timing van het geheel. Hoe vaak kijk je naar inputveranderingen?
Hint: probeer de code kort (en generiek) te houden en niet te vervallen in een lange reeks van if-then-else-en. Je mag i.p.v. de krachtsensor een andere sensor gebruiken wat leidt tot een hoek-VU-meter of licht-VU-meter.

b) Plaats de code voor het afbeelden van een digitale waarde op de LED-module in een functie met het volgende interface:
`void writeToLedModuel(int val)  // met 0 <= val <= 63 (6 bits)`
De functie produceert alleen uitvoer, er wordt geen resultaat opgeleverd. Vandaar het `void`. Pas de grootte van `val` aan aan het aantal beschikbare bits (63 voor 6 bits, 31 voor 5 bits). Roep de functie aan vanuit je meetcyclus.

16. **De Night Rider**

Schrijf een programma waarmee je een "Night Rider"-functie realiseert. Als je niet weet wat dat is, omdat je in je jeugdjaren de populaire TV-serie gemist hebt (en de herhaling ook) "google" het dan even. Maak gebruik van 6 bits (op de LED-module) en maak de snelheid van het veranderende lichtpatroon (looplicht) afhankelijk van de input van een analoge sensor.
Merk op dat er bij de echte "Kitt" meerder bits bewegen. Kies voor 2 bewegende bits over een bereik van 6.
[Optioneel] Om het Night Rider-effect nog realistischer te maken kun je proberen gebruik te maken van de PWM-functie die sommige pinnen bieden door LED's te laten opgloeien. Doe dit laatste alleen als je tijd over hebt of als eerbetoon aan David Hasselhoff (die zulke zware jaren heeft gehad in de Betty Ford-kliniek).

17. **De webserver**

```
#include <servo.h>
Servo servo1;     // new servo
servo1.attach(9); // servo on pin 9

servo1.write(position);
delay(15);
```

En dan nu: de "uitsmijter" van deze introductie in embedded systems en embedded programming: de Arduino met IO-webserver.

Inleiding:
Als onderdeel van de Arduino software worden diverse software-componenten meegeleverd in de vorm van specifieke libraries (documenten -> arduino -> libraries). Controleer dat. Standaard wordt er, bijvoorbeeld, een servo-library, meegeleverd waarmee het werken met servo's sterk wordt vereenvoudigd. Zie kader en opdacht 18.

Op vergelijkbare wijze kan voor de ethernet-shield een webserver (component) op poort 80 worden geconfigureerd:

```
#include <Ethernet.h>
```

```
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 1, 3);

EthernetServer server(80);  // start http server on port 80
Ethernet.begin(mac, ip);
server.begin();
```

Gevraagd/Doe:

- Plaats de Arduino ethernetshield op de Arduino UNO en download het bijbehorende programma (Arduino script, iowebserver.ino) via blackboard en laadt het in je Arduino.

- Sluit je ethernetshield aan op je laptop of (beter nog) op een router, waar uiteraard ook je laptop aan hangt (bedraad of draadloos). Onderzoek wat het (bedrade) IP-adres van je laptop is en noteer dit. Merk op dat je laptop naast de bedrade aansluiting ook nog verbonden kan zijn met het internet via de draadloze verbinding. Dat is handig, want dan kun je ondertussen stackoverflow raadplegen. Een directe (doorgaans bedrade) verbinding tussen twee systemen noemt men een ad hoc-netwerk. In beide gevallen, ad hoc of via een router, is het verstandig de netwerkadressen te kiezen in het segment 192.168.1.x, waarbij x het computernummer is. Het voordeel van dit segment is dat het strikt lokaal is. De informatie van computers in dit segment worden niet doorgegeven over het internet.

- Bestudeer het webserver-programma en onderzoek of je webserver een IP-adres krijgt en zo ja, welk? Gebruik de Serial Monitor. Noteer dit adres ook en vergelijk het met dat van je laptop. Zitten beide systemen in hetzelfde netwerk(segment)? De toekenning in de Arduino kan statisch of dynamisch, via DHCP. Welke gebruik je? Zorg er hierbij voor dat je in een gecontroleerde netwerkomgeving werkt, met bijvoorbeeld een eigen router waarvan je de instelling kunt aanpassen. Op het NHL-netwerk hebt je weinig vat, wat aanleiding kan vormen tot problemen (met firewalls etc.). Bij de statische configuratie bepaal je zelf wat het IP-nummer is. Je kunt het (in Windows) aanpassen via het configuratiescherm. Een nadeel van een statisch IP is dat je, in een netwerk met meerder computer, conflicten kunt krijgen. Beter is het dan ook je Arduino via DHCP een IP-nummer te laten toekennen. Dit wordt ondersteund door de serversoftware.
  Voor de rest van deze opgave gaan we uit van de volgende configuratie: de laptop krijgt adres 192.168.1.2 en de Arduino(server) 192.168.1.3. Dat kan bij jou dus, zeker als je DHCP gebruikt, anders zijn.

- Probeer, nu je begrijpt hoe eenvoudige netwerkcommunicatie, op het niveau van de IP (-nummers), werkt, de webpagina die is voorgeprogrammeerd in de Arduino-webserver, via je browser zichtbaar te maken door de volgende URL in de voeren:
  http://192.168.1.3
  Nevenstaande site moet zichtbaar worden. Mocht dit niet gebeuren en je weet zeker dat alle IP-adressen kloppen, lees dan verder bij de opmerking onderaan deze opdracht.

Inleidende opdrachtjes

Breng eenvoudige modificaties aan in deze webpagina, zodat je “vat krijgt” op de code. Beperk je voorlopig tot de HTML, door bijvoorbeeld enkele STYLE-elementen aan te passen. Merk het volgende op:

- Het verzoek dat de client (laptop) naar de server (Arduino) stuurt wordt ingeleid met een http-header, een zogenaamd http-request. In deze header zit niet alleen informatie "verstopt" over het protocol zelf, maar ook over de afzender (192.168.1.2), de gevraagde pagina en eventuele variabelen die de client meestuurt. De basis van het verzoek is de vraag naar een internetpagina. De reactie van de server wordt eveneens ingeleid met een http-header, de http-reponse, waarna de gevraagde pagina, vaak "ingepakt" in HTML-codes, wordt teruggestuurd naar de client. Dat is de manier waarop computers via het HTTP-protocol over internet met elkaar communiceren. Deze embedded code biedt dus een aardig kijkje onder de internet-motorkap. Doe er je voordeel mee.
- Deze communicatie is niet alleen gekoppeld aan de beide IP-nummers, maar ook aan een poort. Voor HTTP is dat altijd poort 80. Onderzoek in de code waar de server wordt gestart en de toekenning van het IP-address en de IP-port wordt gedaan.
  Meer informatie over het HTTP-protocol:
  `http://betterexplained.com/articles/a-simple-introduction-to-computer-networking/`
  `http://code.tutsplus.com/tutorials/http-the-protocol-every-web-developer-must-know-part-1--net-31177`
- De theorie en de vaardigheden, zoals geleerd bij het vak Web Development, kunnen hier toegepast worden. Uiteraard zijn er enige beperkingen i.v.m. de kracht van de Arduino in termen van processor performance en geheugengrootte. Helaas ondersteunt de Arduino geen PHP en ook geen ASP.NET, wel (natuurlijk) allerlei client-side (Javascript) software.
- Wat kun je doen als je via je laptop/browser geen contact krijgt met je Arduino-server en je je IP-configuratie gecontroleerd hebt en ook begrijpt? Onderzoek welke variant van het ethernet-shield je hebt. Er zijn namelijk, naast enkele incourante types, twee versies in omloop met de naam "Ethernet shield" en "Ethernet shield 2", wat meestal wel op het Arduino-bordje staat of op het doosje. Toen "Ethernet Shield" op de markt kwam was het nog niet bekend dat er ook een "Ethernet Shield 2" zou komen, vandaar de naamgeving. Bij gebruik van "Ethernet shield" (versie 1 dus) moet het contact tussen laptop en Arduino geen probleem zijn. Bij "Ethernet shield 2", is het aan te bevelen de bijbehorende library te installeren. Deze draagt ook de naam "Ethernet2" (wat wel goed gekozen is;). Meer informatie over dit nieuwe shield, waaronder ook de manier waarop je libraries installeert (eigenlijk gewoon kopieert) tref je aan door de onderstaande links te bekijken.
  http://www.arduino.org/products/shields/5-arduino-shields/arduino-ethernet-shield-2
  http://labs.arduino.org/Arduino%20Ethernet%20Shield%202
  Nog enkele technische details
  - het nieuwe shield is gebaseerd op een nieuwe ethernet controler, de W5500-chip.
    Deze is in de plaats gekomen van de W5100.
  - In je (voorbeeld)code moet met de volgende aanpassing rekening houden:
  #include <Ethernet.h> //If you are using Ethernet Shield 2, use instead <Ethernet2.h>

  Sluit een sensor aan op je Arduino (bijvoorbeeld de krachtsensor op kanaal 0) en modificeer je programma zodanig dat je uitsluitend de sensorwaarde, niet de hele string "Analog sensor (channel 0): ", bold afdrukt en tevens rood indien deze boven de 600 komt. Onderzoek tevens de mogelijkheid om dit (aan de client-kant) automatisch herhaald te doen, met een update om de 3 seconden. Hint: Meta Tag. Gebruik zo nodig:
  `http://www.w3schools.com/html`
  `http://www.w3schools.com/cssref/css_colornames.asp`

Hoofdopdracht

- Modificeer je programma zodanig dat je via een GET-variabele, als onderdeel van je URL, je server opdracht kunt geven de afzonderlijke LED's aan de pinnen p8 en p9 aan en uit kunt zetten. In principe zou je dit kunnen uitbreiden voor alle pinnen (p2 .. p13), maar aangezien p10..p13 wordt gebruikt voor de besturing van de ethernetshield zelf, beperken

we ons tot p8 en p9. Hanteer het volgende interface:
http://192.168.1.3/?p8=1
Waarmee de LED aan pin 8 aan wordt aangezet. Overeenkomstig kan met
http://192.168.1.3/?p9=0
de LED aan pin 9 worden uitgezet (terwijl die aan pin 8 natuurlijk aan blijft).
Zoek in je code naar het punt waar de URL verwerkt (geparsed) wordt en splits deze in haar onderdelen, argument en value, om er achter te komen welke LED aan of uit moet.
Gebruik in je programmacode de volgende functie:

```
if (parseHeader(httpHeader, arg, val)) {
   digitalWrite(arg, val);                    // switch LED on/off
   client.print("Pin "); client.print(arg);
   client.print(" = "); client.println(val);  // tell the user
}
```

De functie dient als volgt gedefinieerd te worden:

```
bool parseHeader(String header, int &a, int &v)
```

Houd je aan dit interface. Merk op dat de functie een boolean teruggeeft. De betekenis daarvan is dat deze "true" oplevert indien de functie haar werk goed doet en "false" indien dat niet zo is. De parameters `a` en `v` zijn uitvoer-parameters (of reference parameters). Een toekenning, binnen de functie, van een waarde aan één van de beide parameters leidt tot een aanpassing van de argumenten bij aanroep (verander je `a`, dan verandert `arg` dus mee).
Meld de gebruiker, via het webinterface, de mogelijke veranderingen in de status van de beide pinnen, ook als er zich fouten voordoen.

Merk op:
- dat er geen specifieke webpage wordt opgevraagd, vandaar het vraagteken direct na de forward slash.
- GET-variabelen kunnen via de URL worden doorgegeven in paren van argument en value, gescheiden door een puntkomma. Hier is er sprake van maar één paar, nl px = v, waarbij x = 8 of 9 en v = 0 of 1. Onderzoek bij de parsing van de header het gebruik van `indexOf()`. Zoek naar "?" en "=".
- Goeie programma's kenmerken zich o.a. door robuustheid. De server en webpage blijven goed werken ook als de gebruiker (per ongeluk) invoert: http://192.168.1.2/?p13=6 of http://192.168.1.2/?p8=aan

Tot slot:
Deze opdracht plaatst programmeren niet alleen in een embedded context, ook de concepten rondom internet- en netwerkenprogrammering en client-server-programmering, komen aan de orde. De opdracht is dan ook de "perfecte" opmaat voor het domotica-project. Daar gaan we de functionaliteit van de (embedded) website vervangen en uitbreiden met een (native) app voor een smartphone. Het http-protocol wordt daarbij ingewisseld voor communicatie m.b.v. sockets. Stay tuned.

Veel plezier met deze **opdracht** en het navolgende **Domotica-project**. Verdieping:
[Optioneel]
- Breid je code uit met HTML-Forms en plaats knopjes op je scherm waarmee de pinnnen 8 en 9 aan en uitgezet kunnen worden.
- Sluit, ter voorbereiding op het Domotica-project, je RF-zender al aan en probeer via een webinterface en je klik-aan-klik-uit-device de verlichting de regelen.

- Plaats een SD-card in je ethernetshield en laadt de webpagina van SD-card.
- Breng de Arduino ter sprake als de gesprekken, thuis aan tafel of in de kroeg stil vallen . Doe dit met dermate veel enthousiasme dat vriendinnen, jongere broertjes en neefjes ook informatica willen gaan studeren ;)

18. **De servomotor**

In het Domotica-project wordt, bij de laatste (C-)opdracht, een (mini)servo-motor gebruikt. Een servo zet een getal uit de Arduino, via een elektrisch signaal, om naar een hoekpositie. Gebruik de eigen servo die je ter voorbereiding op deze module hebt aangeschaft. Bevestig, om de hoekuitslag goed te kunnen waarnemen, een van de bijgeleverde hulpstukjes op het asje van het motortje en verbind daaraan een potlood, bijvoorbeeld met een tapeje of elastiekje
Sluit de servo aan op een PWM-pin naar keuze, bijvoorbeeld pin 9. Vanuit het perspectief van de software is een servo een component die als volgt gebruikt kan worden:

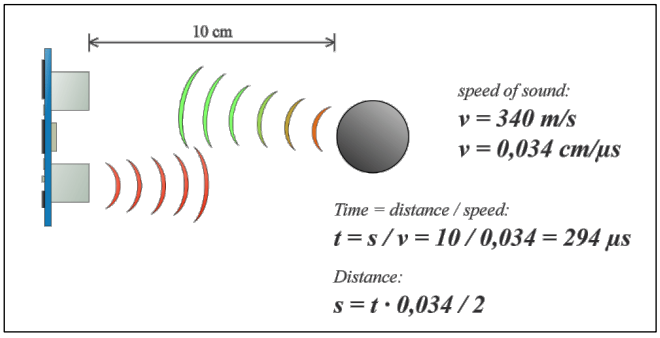
```
#include <servo.h>    // header file voor de servo-component
Servo servo1;         // new servo
servo1.attach(9);     // servo on pin 9
```

Het is aan jullie zelf om verder uit te zoeken dit werkt en ook hoe de hoekpositie van de (fysieke) servo aangepast kan worden, ook periodiek. Er zijn talloze voorbeelden, tutorials en video's te vinden. Bijvoorbeeld: https://www.youtube.com/watch?v=BfMfysmfoNM

Pas op: Een mini-servo kan gevoed worden vanuit de Arduino (5V en GND), ook als deze op haar beurt gevoed wordt vanuit de USB van je laptop. Gebruik in deze configuratie geen grotere of zwaarder servo's. Deze nemen aanzienlijk meer stroom (en vermogen) waardoor er problemen kunnen ontstaan met (de voeding vanuit) je laptop.

Schrijf een programma waarmee je de stand van (de rotatie-as van) de servomotor kunt laten variëren met behulp van een sensor naar keuze. Je kunt de krachtsensor gebruiken, maar een sensor die een ander grootheid meet, zoals licht of temperatuur, mag ook. Zorg dat je de schalen van sensor en servo op elkaar afstemt, waarbij een minimaal niveau (geen kracht/licht) leidt tot geen uitslag en een maximaal niveau (maximaal kracht/licht) tot een maximale uitslag. Onderzoek hierbij het gebruik van de functie `map()`

19. **De ultrasone afstandssensor**



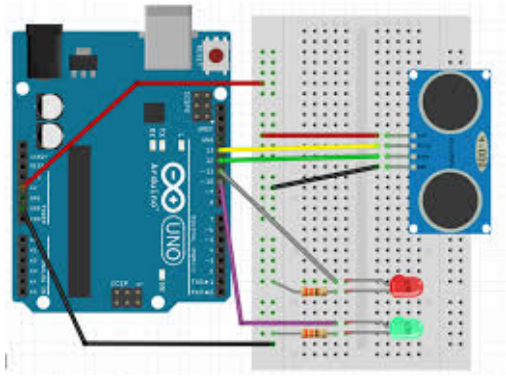
Het meten van afstand kan op verschillende manieren. De afstandssensor die bij het Domtica-project wordt gebruikt, en hier wordt geïntroduceerd, werkt volgens met ultrasone geluidsgolven. Het interface is digitaal, met duidelijk omschreven signaallijnen en bijbehorend protocol. Dit protocol is gebaseerd op het meetprincipe wat is afgekeken van de vleermuis, nl. het bepalen van een afstand(sbeeld) op basis van reflecties van (ultrasone) geluidsgolven.

De sensor heeft 4 pinnen, de buitenste voor + (Vcc) en – (Gnd) en de binnenste voor Trig en

Echo. Op grond van die benaming is, zeker in combinatie met de verwijzing onder aan deze opdracht, te begrijpen dat via de signaal Trig (aan een Arduino output) een korte puls wordt gegeven en dat de reflectie van deze puls, via het object tot waar de afstand wordt gemeten, binnenkomt bij Echo (aan een Arduino input). Via een tijdsmeting wordt de afstand berekend. Geluid plaatst zich immers met een (nagenoeg) constante snelheid voort door de lucht.
De afzonderlijke stappen die nodig zijn voor het uitvoeren van een meting moeten in software worden vastgelegd. Het is aan eenieder om dit zelf te bedenken of op te zoeken. Er is voldoende materiaal en voorbeeld code beschikbaar, in tekst en video.

a) Omschrijf de werking van de ultrasone sensor in eigen woorden, inclusief het natuurkundig principe waarop het gebaseerd is. Wat is de betekenis van het getal 340 in de formule voor de tijdsbepaling en waarom moet er door 2 worden gedeeld? Wat zijn (bij benadering) de minimale en de maximale meetafstand en wat is de nauwkeurigheid (in cm.)?

b) Schrijf een programma waarmee je periodiek de afstand tot een object meet en deze toont via de Serial Monitor. Kies als meetinterval 250 msec.
Pas op: Wanneer er zich geen object in de bundel bevindt, wordt er geen (ultrasone) puls gereflecteerd. Hoe stel je dit vast in je programma? Welke afstand of status hoort hierbij? Hoe reageert je programma hierop?



c) Schrijf een functie met het volgende interface:
`int distance(int trig, int echo)`
Deze functie krijgt twee parameters aangeleverd, nl. de pinnummers voor de beide signaallijnen. Daarmee ontstaat een flexibele koppeling tussen functie en pinnummers en kan er op eenvoudige wijze een tweede ultrasone sensor worden aangesloten. De functie levert een afstand in cm op en -1 wanneer er geen afstand gemeten kan worden, met een time out op grond van geen reflectie. Kies de timeout waarde strategisch (zodat het niet te lang duurt), bijvoorbeeld bij metingen boven de 200 cm.

d) Schrijf een functie met het volgende interface:
`int distanceAv(int trig, int echo)`
Deze functie maakt gebruik van `distance` en bepaalt het gemiddelde van 6 metingen, waarbij de hoogste waarde en de laagste waarde worden genegeerd en de resterende vier gemiddeld. Dit geeft een betrouwbaardere afstand, met als keerzijde dat het meer tijd neemt (let op de time out).

e) Schrijf een programma waarmee je de gemeten afstand koppelt aan de VU-meter van opdracht 15, met alle LED's uit bij minimale afstand, alle LED's aan bij maximale afstand en een alarmerende knippering gedurende 1 sec., met afwisselend alle LED's aan en uit, bij een foute meting (= geen reflectie). Kies hierbij een frequentie van 10 Hz, dus 10 keer afwisselend aan en uit.

refr.
https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/
https://www.youtube.com/watch?v=0Aj5VwhZT8Q

20. [Optioneel] Combinatie-opdracht met servomotor, afstandssensor en webserver.
    Als je de servo opdrachten heb uitgewerkt, ligt het voor de hand beide devices te combineren met de webserver van opdracht 17.
    a) Schrijf een programma, als uitbreiding van opdracht 17, waarmee je de servo via een webinterface kunt bedienen. De functie en bediening mag je zelf kiezen. Je kunt natuurlijk GET-variabelen gebruiken, maar je kunt ook experimenteren met specifieke user interface-componenten. Denk aan een slider.
    b) Breid het programma zodanig uit je de vormgeving van de pagina kunt relateren aan afstandsinformatie. Een mogelijke toepassing: plaats de sensor naast je beeldscherm, in de richting van je gezicht. Naarmate de afstand tot je gezicht groter is, wordt de fontgrootte groter, waardoor je de site op afstand nog steeds goed kunt lezen.

21. [Optioneel] Schrijf een programma waarmee met twee personen een reactiespel gespeeld kan worden. Gebruik hiervoor twee krachtsensoren. De werking van het spel:
    - Genereer (random) een 6 bits bitpatroon (het spelpatroon) op de LED's, waarbij er minstens 1 LED aan is.
    - Laat de LED's in dit patroon branden en wacht tot beide spelers de krachtsensor ingedrukt houden (kies hierbij een aantrekkelijke/speelbare kracht).
    - Zodra de laatste speler de sensor loslaat verdwijnt dit patroon en genereert de software periodiek een random 6 bits ledpatroon, met ook nu weer minstens 1 LED aan. Kies een prettige/speelbare duur waarmee de LED's branden en de patronen zich herhalen.
    - Terwijl het random wisselen van de patronen houden de spelers de handen aan de koppen. Zodra het (initieel gegenereerde) spelpatroon verschijnt, is de speler die dit het eerste herkent en dit bevestigt met het indrukken van de krachtsensor de winnaar. Uiteraard dient het patroon waarbij gedrukt is vergeleken te worden met het initiële patroon. Dit moet uiteraard kloppen. In de "wintoestand" wordt het winnende patroon knipperend getoond. Kies prettige/speelbare tijden en krachten.