Ing Patrick Van Houtven

[E-mailadres]

Geluiden/signalen met arduino

DSP

Labo-opdracht A2

# 

# Labo 04 : Geluiden/signalen met arduino

2Ea klasgroep :

Dit labo werd gemaakt door :

Simulatie werd uitgevoerd door student :

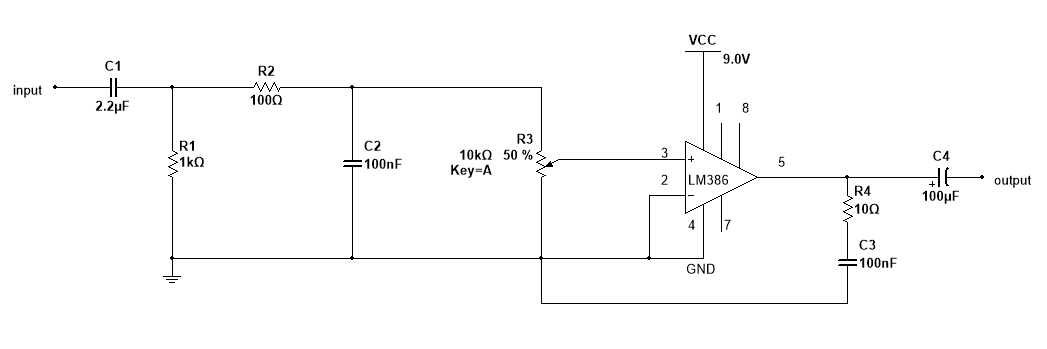
Metingen werden uitgevoerd door

* Student :
* Student :

Dit labo nam … sessies in beslag

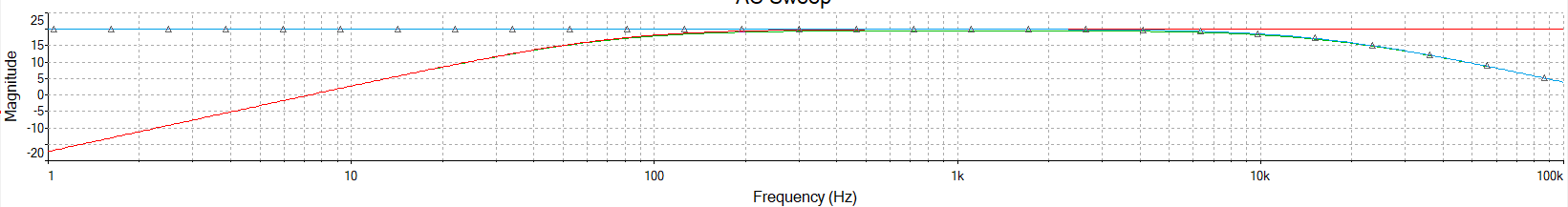
# Opgave 1: karakteristiek van de hardware

***Gegeven onderstaande versterker met RC-filter en gebruik deze om de tonen te kunnen afspelen voor de volgende opgaven in deze labosessie.***



Opdracht : Teken de filterschakeling C1-R1-C2-R2 van bovenstaand schema in multisim en voer hierop een AC-analyse uit. Neem als output de spanning over C2. Als je niet over de juiste componentwaarden beschikt probeer je een samenstelling te maken zodat de tijdsconstante van C1R1 zo goed mogelijk dezelfde waarde blijkft. Hetzelfde kan je doen met de RE-C2 combinatie.

Voorbeeld van AC-analyse:



✂️*Plaats een snippet van het R1-C1-R2-C2-gedeelte in multisim hier.*

✂️*Plaats een snippet van de AC-sweep van R1-C1-R2-C2 gedeekte*

Graphical user interface, text, application

Description automatically generated

|  |  |
| --- | --- |
| **Vraag 1** | **Deze schakeling wordt aan de PWM-uitgang van arduino aangesloten. Verklaar de werking van het *R1-C1-R2-C2-*netwerk. Wat is de functie van deze schakeling?** |
| **Antwoord** | De RC filter is een banddoorlaatfilter en wordt hier gebruikt om ruis en uitschieters te verzwakken. |

Bouw de volledige schakeling op een klein breadboard op en laat deze opgebouwd zodat je die bij volgende labosessies ook nog kan gebruiken. Als je de versterker op een PCB-hebt via PCB-design kan je die ook gebruiken.

📷 Neem een foto van de gemaakte opstelling op breadboard. Tezamen met de leesbare studentenkaarten van de groepsleden.



Meet de maximale onvervormde versterking op bij 1 kHz (sinus) van bovenstaande versterkerschakeling en druk deze uit in dB.

📷 Neem een foto van het in- en uitgangssignaal weergegeven op een scoop

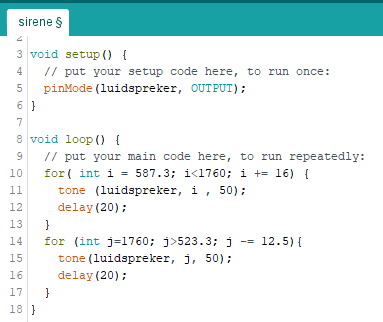


|  |  |  |
| --- | --- | --- |
|  | Ch1 (ingang) | Ch2 (uitgang) |
|  | **5V** | **9.18V** |
| frequentie | **1kHz** | **1kHz** |

|  |  |
| --- | --- |
| **Vraag 2** | **Bereken hier de spanningsversterking in dB** |
| **Antwoord** | 20log(9,18V/5V)=5,28dB |

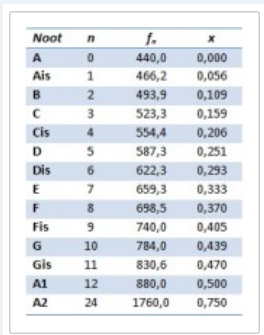
# Opgave 2: akoestisch alarm

***Voer de sketch “eenvoudig en intermitterend akoestisch alarm” uit en gebruik de LM386-versterker om de luidspreker aan te sluiten op arduino.***

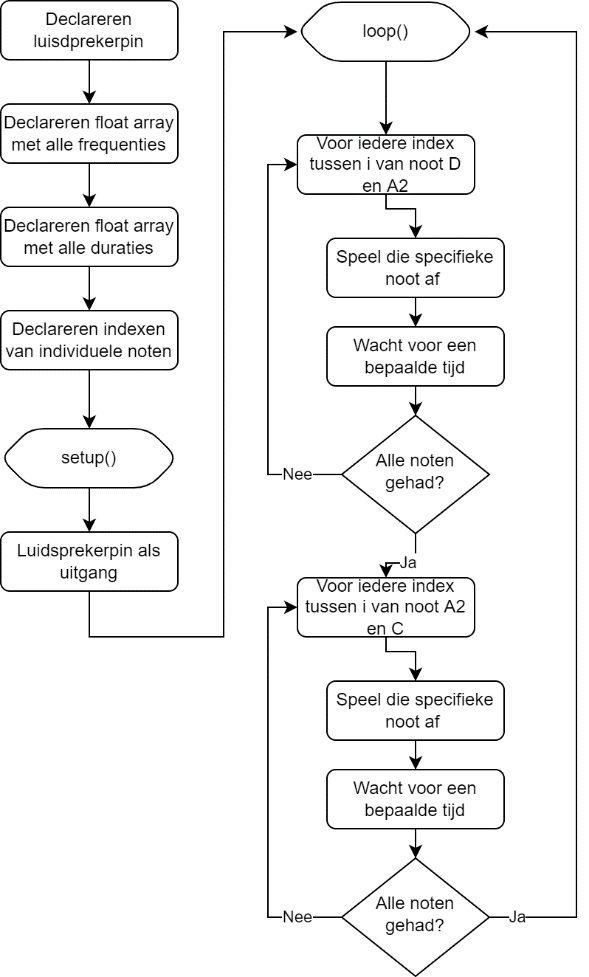


|  |  |
| --- | --- |
| **Vraag 3** | **Beschrijf wat er gebeurt als je bovenstaande sketch runt in combinatie met de hardwareschakeling.** |
| **Antwoord** | Bovenstaande sketch genereert een alarmsignaal lijkend op dat van bijvoorbeeld de politie. |
| **Vraag 4** | **Verklaar wat de tone()-functie doet en welke parameters je deze functie kan meegeven.** |
| **Antwoord** | De tone()-functie genereert een blokgolf op een gespecifieerde frequentie. De eerste parameter is de pin waarop we het signaal willen genereren, de tweede parameter is de frequentie en de laatste parameter specifieert de duratie maar is optioneel. |

***Pas bovenstaande sketch aan zodat de µC het alarm laat oplopen startend van de D-toon tot A2-toon en laat teruglopen in toonhoogte van A2-toon tot C-toon. De tijdsduur voor het oplopen bedraagt ongeveer 2 seconden en het teruglopen ongeveer 1,5 seconden.***



***? Geef grafisch weer (soort van stroomdiagram) hoe je het alarm aanpast met de gevraagde toonhoogte***

******

|  |  |
| --- | --- |
| **Vraag 5** | **Verklaar de werking van je geschreven code aan de hand van het hierboven getekende stroomdiagram** |
| **Antwoord** | In de code wordt gebruik gemaakt van 4 arrays. Twee arrays zijn er specifiek om voor iedere noot de frequentie (f[]) en duratie (x[]) op te slaan. Daarna wordt er, voor de gebruikte noren gedefinieerd op welke index de noot plaatsvindt en krijgen ze een alias bv.: “D=5”.  Deze indexen worden in een for loop gebruikt om alle tussenliggende noten in f[] ook af te spelen. Eerst van D naar A2 en dan van A2 naar C |

|  |  |
| --- | --- |
| **CODE!** | **Plaats hier je geschreven code** |
| // Akoestisch Alarm  // Jesse Denaux & Daan Dekoning Krekels  #define LUIDSPREKER 2  const float f[] = {440.0, 466.2, 493.9, 523.3, 554.4, 587.3, 622.3, 659.3, 698.5, 740.0, 784.4, 830.6, 880.0, 1760.0}; // A -> A2 : frequentie  const float x[] = {0.000, 0.056, 0.109, 0.159, 0.206, 0.251, 0.293, 0.333, 0.370, 0.405, 0.439, 0.470, 0.500, 0.750 }; // A -> A2 : tijd  #define C 3  #define D 5  #define A2 13  void setup() {  pinMode(LUIDSPREKER, OUTPUT);  }  void loop() {  for(byte i = D; i <= A2; i += 1){  tone(LUIDSPREKER, f[i], x[i]);  delay(2000/(A2-D));  }  for (byte j=A2; j >= C; j -= 1){  tone(LUIDSPREKER, f[j], x[j]);  delay(1500/(A2-C));  }  } | |

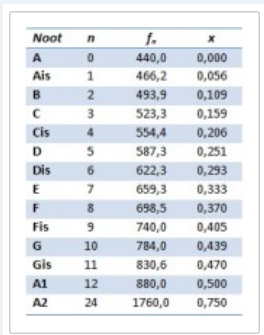
# Opdracht 3: creëer een liedje

***Timing van het geluid is belangrijk om een herkenbaar muziekje te kunnen maken. Vervang de sketch zodat de controller het liedje broeder Jacob afspeelt.***

***De sequentie voor de noten is :***

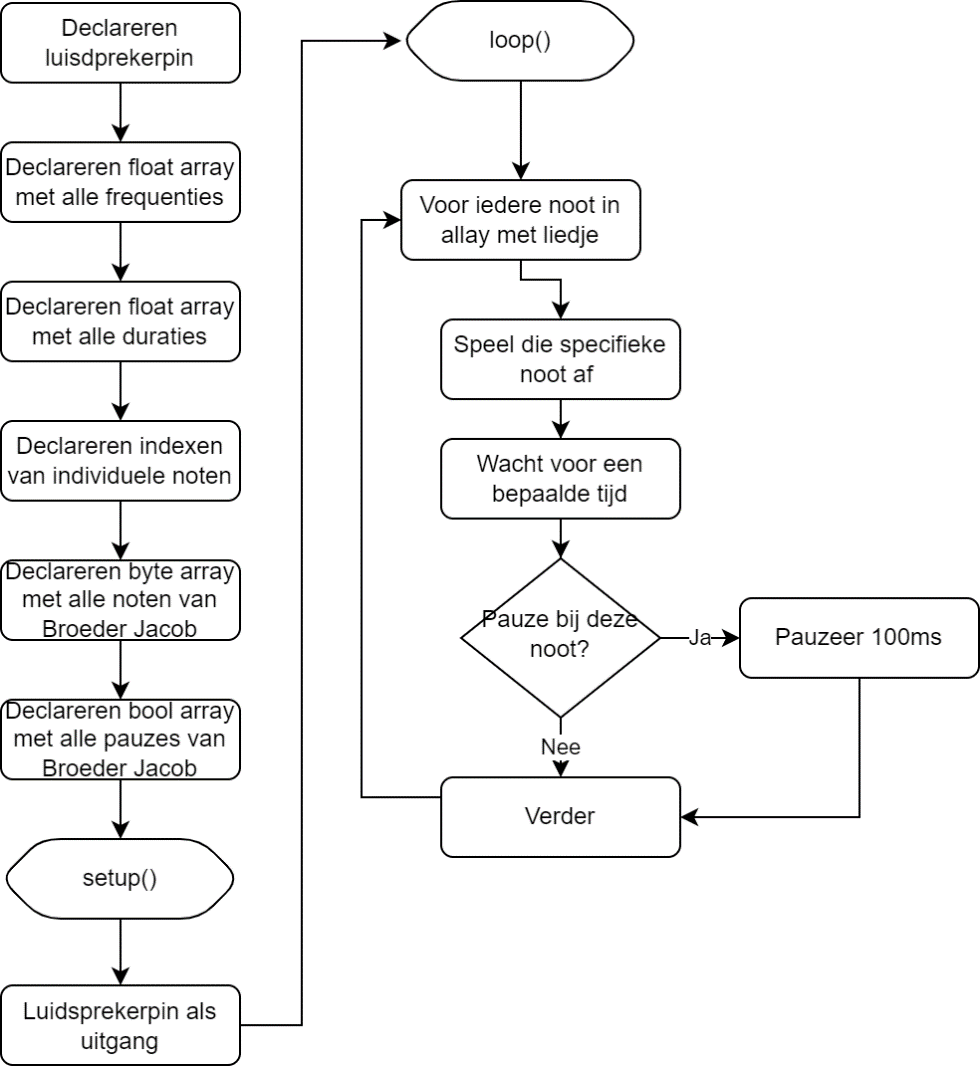
***CDEC CDEC DEF DEF FGFED C FGFED C C G(392 Hz) C C G(392 Hz) C***

***Onderstaande figuur geeft de overeenkomstige toonfrequenties aan voor de desbetreffende muzieknoot.***



***Pas de duur van de tonen aan het liedje aan en voeg wachttijden in daar waar nodig (artistieke vrijheid is toegestaan ☺ )***

***? Geef grafisch weer (soort van stroomdiagram) hoe je het liedje broeder Jacob laat afspelen zodat het herkenbaar in de oren klinkt.***



|  |  |
| --- | --- |
| **Vraag 6** | **Verklaar de werking van je geschreven code aan de hand van het hierboven getekende stroomdiagram** |
| **Antwoord** | In de code wordt gebruik gemaakt van 4 arrays. Twee arrays zijn er specifiek om voor iedere noot de frequentie (f[]) en duratie (x[]) op te slaan. Daarna wordt er, voor de gebruikte noren gedefinieerd op welke index de noot plaatsvindt en krijgen ze een alias bv.: “D=5”.  Twee andere arrays bewaren het af te spelen liedje. Een daarvan (liedje[]) bestaat uit een sequentie van de eerder gedefinieerde noten. De laatste array (snelh[]) bepaalt voor iedere noot of er na het afspelen ervan een korte pauze moet plaatsvinden.  In de loop() gaan we alle noten in de array met het lied in af. Deze array bestaat dus uit de indexen van de specifieke noten dus we kunnen telkens de noot afspelen. Na het afspelen van de noot wordt nog gecontroleerd of er na de noot nog een pauze moest zijn, zo ja dan zal er nog een delay() volgen. |

|  |  |
| --- | --- |
| **CODE!** | **Plaats hier je geschreven code** |
| // Liedje maken  // Jesse Denaux & Daan Dekoning Krekels  #define LUIDSPREKER 2  const float f[] = {440.0, 466.2, 493.9, 523.3, 554.4, 587.3, 622.3, 659.3, 698.5, 740.0, 784.4, 830.6, 880.0, 1760.0, 392.0}; // A -> A2 : frequentie  const float x[] = {0.000, 0.056, 0.109, 0.159, 0.206, 0.251, 0.293, 0.333, 0.370, 0.405, 0.439, 0.470, 0.500, 0.750 }; // A -> A2 : tijd  #define C 3  #define D 5  #define E 7  #define F 8  #define G 10  #define Fis 9  #define A 12  #define B 2  #define Gl 14  const byte liedje[] = {C,D,E,C, C,D,E,C, E,F,G, E,F,G, G,A,G,F,E, C, G,A,G,F,E, C, C, Gl, C, C, Gl, C};  const bool snelh[] = {0,0,0,1, 0,0,0,1, 0,0,1, 0,0,1, 0,0,0,0,1, 1, 0,0,0,0,1, 1, 1, 1, 1, 1, 1, 1};  const float pauze[] = {0.25,0.25,0.25,0.25, 0.25,0.25,0.25,0.25, 0.25,0.25,0.5, 0.25,0.25,0.5, 0.125,0.125,0.125,0.125,0.25,0.25, 0.125,0.125,0.125,0.125,0.25,0.25, 0.25,0.25,0.5, 0.25,0.25,0.5};  byte liedje\_length = sizeof(liedje)/sizeof(liedje[0]);  void setup() {  pinMode(LUIDSPREKER, OUTPUT);  }  void loop() {  for(byte i = 0; i < liedje\_length; i++){  tone(LUIDSPREKER, f[liedje[i]], x[liedje[i]]);  delay(pauze[i]\*1000);  if (snelh[i]){  tone(LUIDSPREKER, f[liedje[i]], x[liedje[i]]);  }  }  } | |

Meet met Ch1 de PWM-uitgang van je schakeling en met Ch2 de

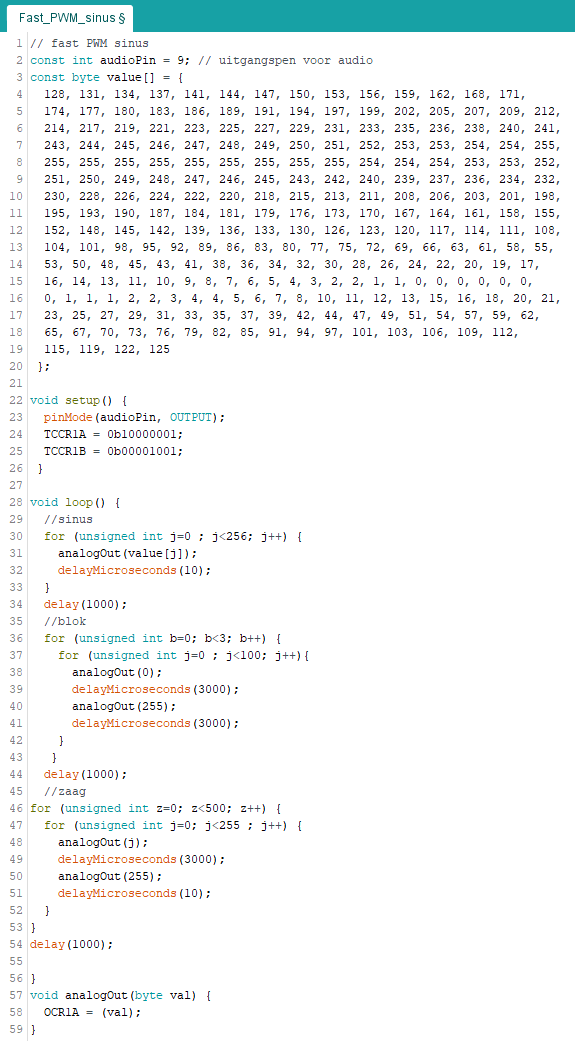
📷 Neem een foto van het in- en uitgangssignaal weergegeven op een scoop



|  |  |  |
| --- | --- | --- |
|  | Ch1 (ingang) | Ch2 (uitgang) |
|  | **5,12V** | **4,48V** |
| frequentie | **521Hz** | **521Hz** |

# Opgave 4 : creatie van sinusgolf, blokgolf en zaagtand

***Gegeven onderstaande sketch voor het genereren van een sinus, blok en zaagtand.***



## De sinusgolf

***? Geef grafisch weer (soort van stroomdiagram) aan de hand van bovenstaande code hoe je een sinusgolf kan genereren vanuit Arduino.***

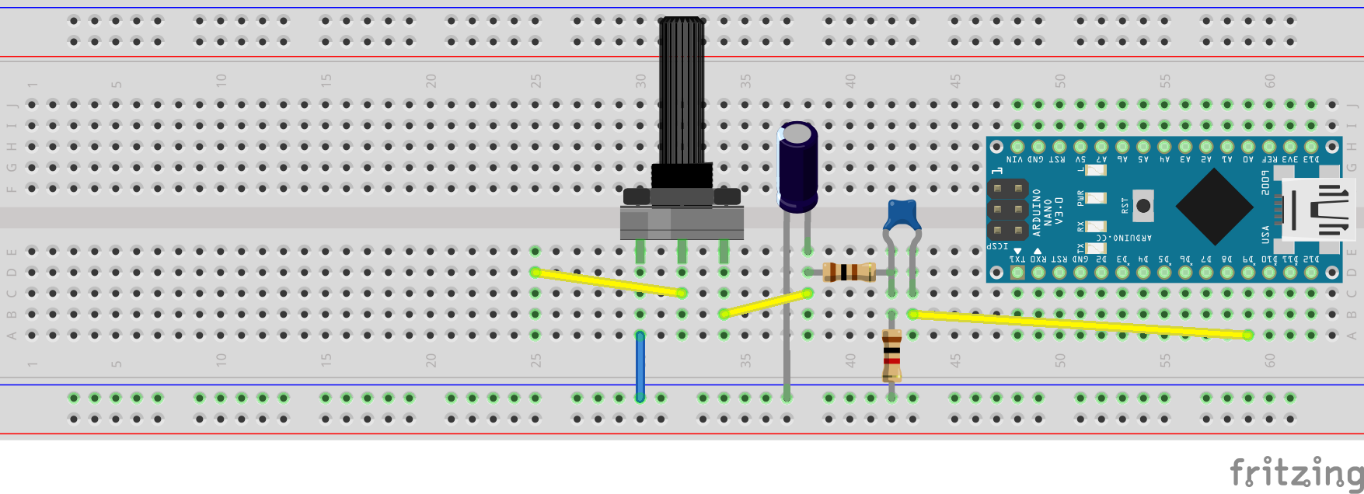
|  |  |
| --- | --- |
| **Vraag 7** | **Verklaar het genereren van het sinussignaal via gebruik van timer 1 en PWM (waarvoor dienen TCCR1A, TCCR1B, OCRIA en de functie analogOut in bovenstaand programma?)** |
| **Antwoord** | TCCR1A en TCCR1B worden gebruikt om TC1 in te stellen.  OCR1A het output compare register voor TIMER1.  analogOut stelt telkens OCR1A in op de waarde in de value[] array. |

Maak een sketch voor enkel het genereren van een sinusgolf aan de hand van bovenstaand voorbeeldsketch.

|  |  |
| --- | --- |
| **CODE!** | **Plaats hier je geschreven code** |
| // fast PWM sinus  const int audioPin = 9; // uitgangspen voor audio  const byte value[] = {  128, 131, 134, 137, 141, 144, 147, 150, 153, 156, 159, 162, 168, 171,  174, 177, 180, 183, 186, 189, 191, 194, 197, 199, 202, 205, 207, 209, 212,  214, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235, 236, 238, 240, 241,  243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 253, 254, 254, 255,  255, 255, 255, 255, 255, 255, 255, 255, 255, 254, 254, 254, 253, 253, 252,  251, 250, 249, 248, 247, 246, 245, 243, 242, 240, 239, 237, 236, 234, 232,  230, 228, 226, 224, 222, 220, 218, 215, 213, 211, 208, 206, 203, 201, 198,  195, 193, 190, 187, 184, 181, 179, 176, 173, 170, 167, 164, 161, 158, 155,  152, 148, 145, 142, 139, 136, 133, 130, 126, 123, 120, 117, 114, 111, 108,  104, 101, 98, 95, 92, 89, 86, 83, 80, 77, 75, 72, 69, 66, 63, 61, 58, 55,  53, 50, 48, 45, 43, 41, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 19, 17,  16, 14, 13, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 2, 1, 1, 0, 0, 0, 0, 0, 0,  0, 1, 1, 1, 2, 2, 3, 4, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15, 16, 18, 20, 21,  23, 25, 27, 29, 31, 33, 35, 37, 39, 42, 44, 47, 49, 51, 54, 57, 59, 62,  65, 67, 70, 73, 76, 79, 82, 85, 91, 94, 97, 101, 103, 106, 109, 112,  115, 119, 122, 125  };  void setup() {  pinMode(audioPin, OUTPUT);  TCCR1A = 0b10000001;  TCCR1B = 0b00001001;  Serial.begin(9600);  }  void loop() {  //sinus  for (unsigned int j = 0; j < 252; j++) {  analogout(value[j]);  delayMicroseconds(10);  }  }  void analogout(byte val) {  OCR1A = (val);  Serial.println(val);  } | |

***Teken in Fridzing (***[***http://fritzing.org/home/***](http://fritzing.org/home/)***) de aansluitingen van Arduino met het RC-filter (filterschakeling voor de versterker (zie opgave 1))***

✂️*Plaats een snippet van het fritzing-schema*

******

Meet volgende signalen op: met Ch1 de PWM-uitgang (ingang van het C1R1R2C2-netwerk, Ch2 de spanning over R1 en Ch3 de spanning over C2

📷 Neem een foto de signalen Ch1, Ch2 en Ch3.



|  |  |  |  |
| --- | --- | --- | --- |
|  | Ch1 (ingang) | Ch2 ( | Ch3 ( |
|  |  |  |  |
| frequentie |  | **16,7kHz** | **25kHz** |

|  |  |
| --- | --- |
| **Vraag 8** | **Verklaar de werking van de hardwareschakeling aangaande het omvormen van het PWM-signaal naar een sinusvorm** |
| **Antwoord** |  |

***Pas de sketch aan zodat je de groots mogelijke frequentie van het sinussignaal kan bekomen.***

|  |  |
| --- | --- |
| **CODE!** | **Plaats hier je aangepaste code** |
|  | |

📷 Neem een foto van het scoop-signaal met de hoogste frequentie die je kan bekomen.

|  |  |
| --- | --- |
|  | Ch1 ( |
|  |  |
| frequentie |  |

***Pas de sketch aan zodat je de laagst mogelijke frequentie van het sinussignaal kan bekomen.***

|  |  |
| --- | --- |
| **CODE!** | **Plaats hier je aangepaste code** |
|  | |

📷 Neem een foto van het scoop-signaal met de hoogste frequentie die je kan bekomen.

|  |  |
| --- | --- |
|  | Ch1 ( |
|  |  |
| frequentie |  |

Bijzondere klanken: kerkklok

Voorbeeldcode om een kerkklok-sound te bekomen:



***? Geef grafisch weer (soort van stroomdiagram) aan de hand van bovenstaande code hoe je een kerkklokgeluid kan genereren vanuit Arduino.***

|  |  |
| --- | --- |
| **Vraag 9** | **Verklaar het genereren van het kerkklokgeluid aan de hand van je stroomschema** |
| **Antwoord** |  |

Voer de code uit.

|  |  |
| --- | --- |
| **Vraag 10** | **Wat is jouw indruk van het geluid?** |
| **Antwoord** |  |

Bij echte percussie-instrumenten, (klokken, drums, cymbalen, triangel, …) neemt de amplitude niet lineair af maar exponentieel. Pas de code aan zodat je op een overtuigende(r) manier deze geluiden kunt imiteren.

Maak een sketch voor het genereren van enkele drumgeluiden.

***? Geef grafisch weer (soort van stroomdiagram) hoe je via arduino een drumgeluid simuleert***

|  |  |
| --- | --- |
| **CODE!** | **Plaats hier je geschreven code** |
|  | |

## De blokgolf

***Schrijf een sketch om een blokgolf van 100 HZ via PWM te creëren aan de hand van de voorbeeldcode onder opgave 4.***

***? Geef grafisch weer (soort van stroomdiagram) aan de hand van bovenstaande code hoe je een blokgolf met een bepaalde frequentie kan genereren vanuit Arduino.***

|  |  |
| --- | --- |
| **Vraag 11** | **Verklaar hoe je een blokgolf van 100 Hz kan genereren aan de hand van je getekende stroomdiagram** |
| **Antwoord** |  |

|  |  |
| --- | --- |
| **CODE!** | **Plaats hier je aangepaste code** |
|  | |

## De zaagtand

***Schrijf een sketch om een zaagtand van 200 HZ via PWM te creëren aan de hand van de voorbeeldcode onder opgave 4.***

***? Geef grafisch weer (soort van stroomdiagram) aan de hand van bovenstaande code hoe je een zaagtand met een bepaalde frequentie kan genereren vanuit Arduino.***

|  |  |
| --- | --- |
| **Vraag 12** | **Verklaar hoe je een zaagtand van 100 Hz kan genereren aan de hand van je getekende stroomdiagram** |
| **Antwoord** |  |

|  |  |
| --- | --- |
| **CODE!** | **Plaats hier je aangepaste code** |
|  | |

Zaagtandgeluid kan de indruk wekken van een snaar aanslaan op een gitaar. Pas de code aan om een meer voller geluid te krijgen (denk aan je codeaanpassing voor percussie-instrument nabootsing)

***? Geef grafisch weer (soort van stroomdiagram) aan de hand van bovenstaande code hoe je via een zaagtand een gitaarklank beter kan imiteren.***

|  |  |
| --- | --- |
| **CODE!** | **Plaats hier je aangepaste code** |
|  | |

|  |  |
| --- | --- |
| **Vraag 13** | **Wat is jouw indruk van het geluid?** |
| **Antwoord** |  |

# Functiegenerator

Ontwerp met een arduino een generator die een variabele blok of sinusfrequente kan genereren tussen 100 Hz en 1 KHz.

* Gebruik een externe potentiometer om deze frequentie te regelen. Bijvoorbeeld als naar de analoge ingang 0 V wordt aangelegd, dan verschijnt er 100 Hz aan de uitgang. Als er 5 V wordt aangelegd dan verschijnt er 1000 Hz. De frequentiestappen tussen twee frequenties kan je kiezen. Bvijvoorbeeld : 100 Hz, 150 Hz, 200 Hz, ….
* Gebruik een dipswitch of andere schakelaar om het signaal te laten switchen van sinus naar blok of omgekeerd.
* Zorg voor een uitlezing naar displat (I2C) en console. Hierin geef je de frequentie weer en de vorm van het signaal : bv 400 Hz Sinus of 800 Hz

Gevraagd:

* De code van de sketch
* Stroomdiagram van de werking van de schakeling
* Aansluitschema (voorkeur via Fritzing)
* De sketsch via een afzonderlijke bijlage