

DSP

Labo-opdracht A2

Geluiden/signalen met arduino

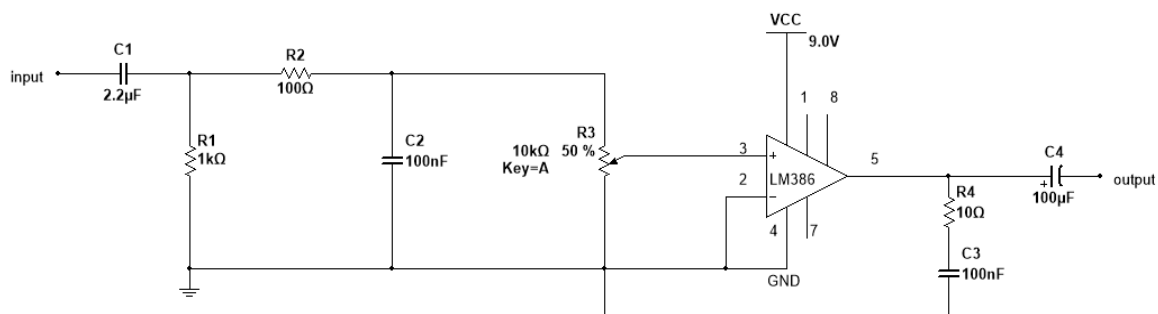
Ing Patrick Van Houtven

s]

Labo 04 : Geluiden/signalen met arduino

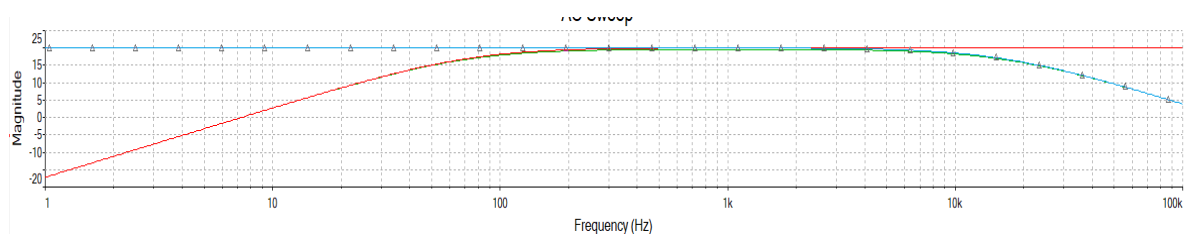
1 Opgave 1: karakteristiek van de hardware

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

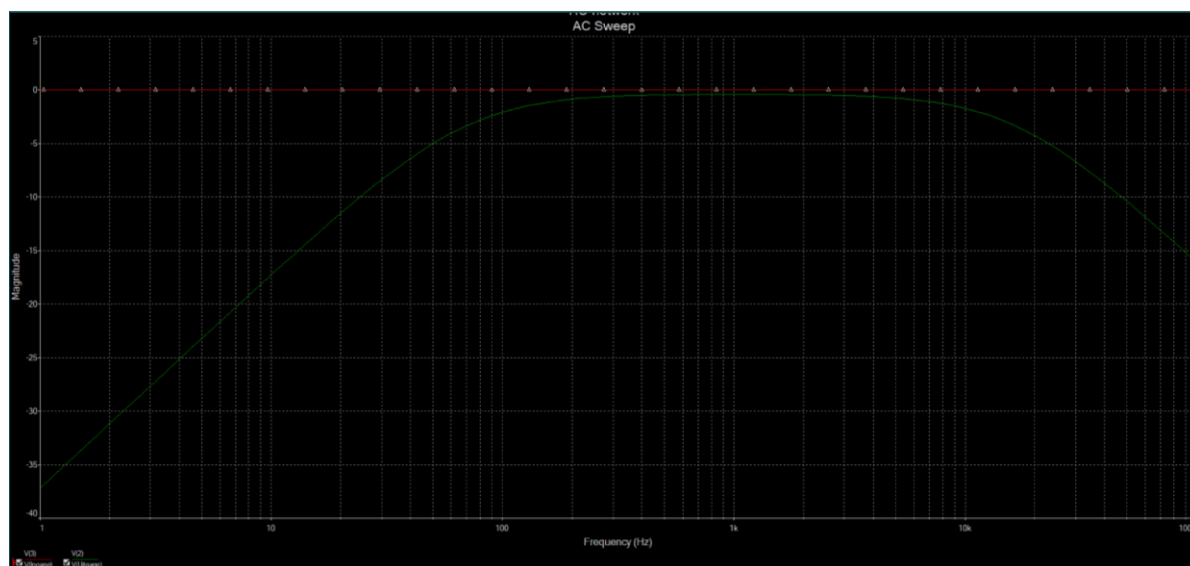



Opdracht : Teken de filterschakeling C1-R1-C2-R2 van bovenstaand schema in multisim en voer hierop een AC-analyse uit. Neem als output de spanning over C2. Als je niet over de juiste componentwaarden beschikt probeer je een samenstelling te maken zodat de tijdsconstante van C1R1 zo goed mogelijk dezelfde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

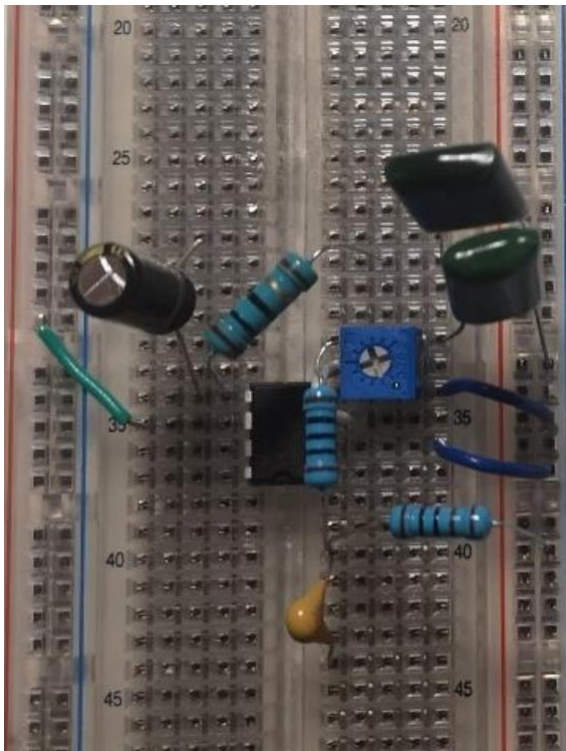


 Plaats een snippet van de AC-sweep van R1-C1-R2-C2 gedeelte

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	Schakeling vormt een bandfilter die enkel het audiofrequentiegebied doorlaat. (75 Hz – 15,9 kHz)

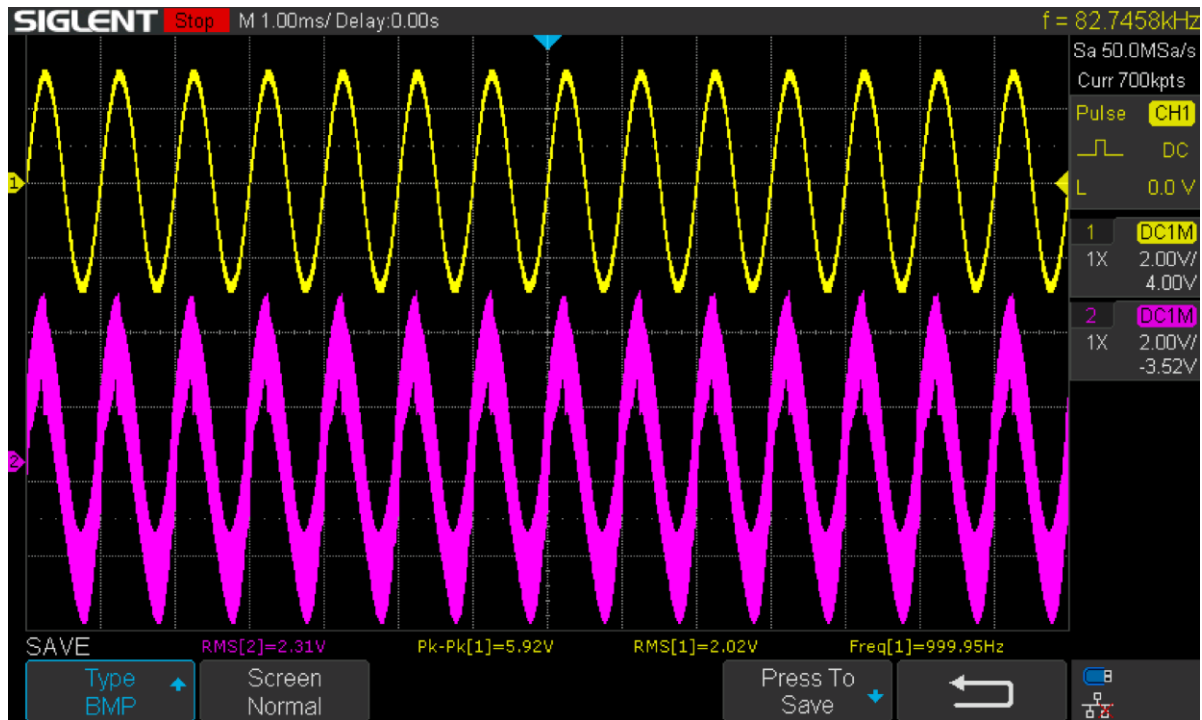
Bouw de volledige schakeling op een klein breadboard op en laat deze opgebouwd zodat je die bij volgende labosessies ook nog kan gebruiken. Als je de versterker op een PCB hebt via PCB-design kan je die ook gebruiken.

 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 scope



	Ch1 (ingang)	Ch2 (uitgang)
$U_{in(max)}$	2,82 V	3,25 V
frequentie	1 kHz	1kHz

Vraag 2	Bereken hier de spanningsversterking in dB
Antwoord	$A_u = 20 \log \frac{U_{out}}{U_{in}} = 20 \log \frac{3,25V}{2,82V} = 1.23 \text{ dB}$

2 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.

sirene \$

```

4
3 void setup() {
4   // put your setup code here, to run once:
5   pinMode(luidspreker, OUTPUT);
6 }
7
8 void loop() {
9   // put your main code here, to run repeatedly:
10  for( int i = 587.3; i<1760; i += 16) {
11    tone (luidspreker, i , 50);
12    delay(20);
13  }
14  for (int j=1760; j>523.3; j -= 12.5){
15    tone(luidspreker, j, 50);
16    delay(20);
17  }
18 }

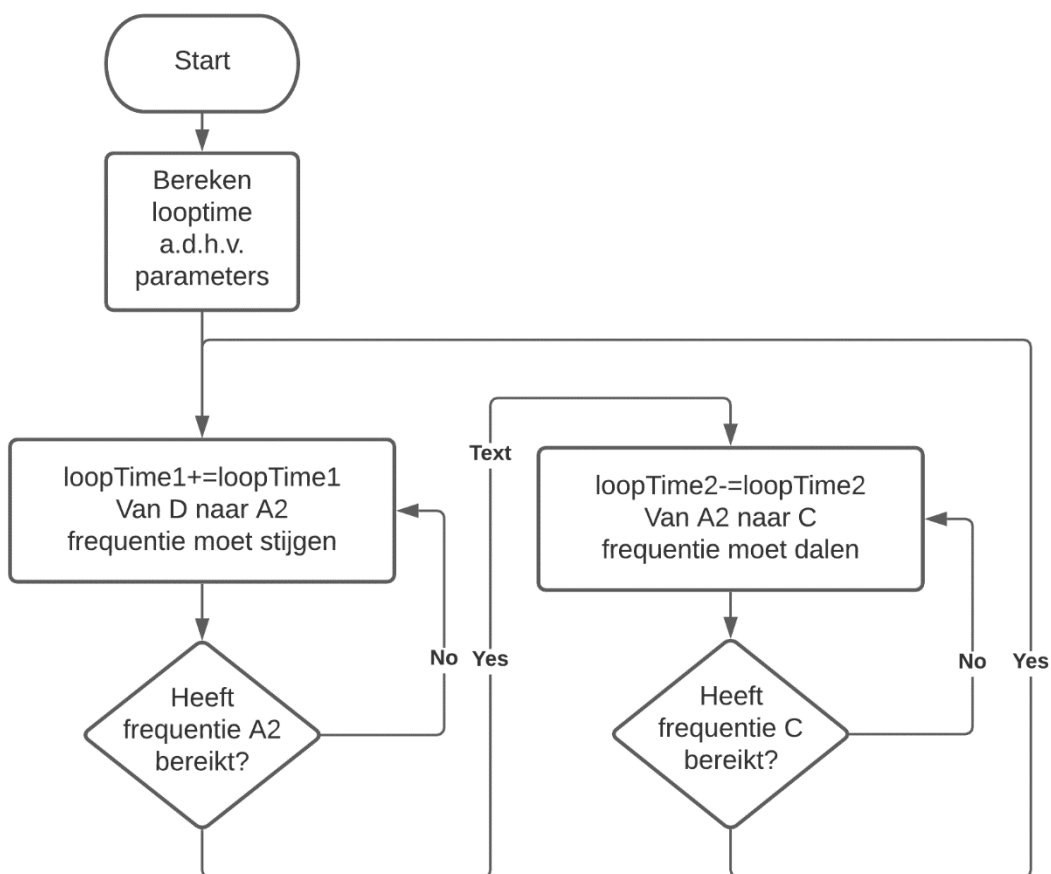
```

Vraag 3	Beschrijf wat er gebeurt als je bovenstaande sketch runt in combinatie met de hardwareschakeling.
Antwoord	Het geluid van een sirene is hoorbaar
Vraag 4	Verklaar wat de tone()-functie doet en welke parameters je deze functie kan meegeven.
Antwoord	De tone()-functie maakt gebruik van hardwaretimers om een bepaalde frequentie te genereren (blokpuls). Het principe is gebaseerd op het afwisselend hoog en laag maken van een bepaalde uitgangspin (duty-cycle 50%). Via een bijkomende parameter kan de tijdsduur van de toon worden gedefinieerd.

Pas bovenstaande sketch aan zodat de μ C 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.

Noot	n	f_n	x
A	0	440,0	0,000
Ais	1	466,2	0,056
B	2	493,9	0,109
C	3	523,3	0,159
Cis	4	554,4	0,206
D	5	587,3	0,251
Dis	6	622,3	0,293
E	7	659,3	0,333
F	8	698,5	0,370
Fis	9	740,0	0,405
G	10	784,0	0,439
Gis	11	830,6	0,470
A1	12	880,0	0,500
A2	24	1760,0	0,750

 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	Eerst wordt de gewenste tijd berekend voor de verschillende stappen die doorlopen worden in de herhalende lussen (For-loop). In de eerste herhalende lus wordt de toon verandert van D naar toon A2. Deze verandering verloopt in stappen van loopTime1. Van zodra de toon A2 is bereikt wordt overgegaan naar de tweede herhalende lus. Hier gebeurt hetzelfde enkel gaan we nu van A2 naar C in een dalende beweging. We gaan dus in stapjes van loopTime2 naar beneden. Van zodra we bij C zijn. Gaan we terug naar het begin van de code. Zo herhaalt het zich.

CODE!	Plaats hier je geschreven code
	<pre> #define luidspreker 4 double loopTime1 = 0; double loopTime2 = 0; int delayWaarde = 10; int time1 = 2000; int time2 = 1500; double fromUp = 587.3; //Van D double toUp = 1760; //Naar A2 double fromDown = 1760 //Van A2 double toDown = 523.3 //Naar C void setup() { pinMode(luidspreker, OUTPUT); loopTime1 = (toUp-fromUp)/(time1/(double)delayWaarde); //Van D naar A2 op 2000mS loopTime2 = (fromDown-toDown)/(time2/(double)delayWaarde); //Van A2 naar C op 1500mS } void loop() { // put your main code here, to run repeatedly: for(int i = fromUp; i < toUp; i+=loopTime1) { tone(luidspreker, i); delay(10); } for(int i = fromDown; i > toDown; i-=loopTime2) { tone(luidspreker, i); delay(10); } } </pre>

3 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 :

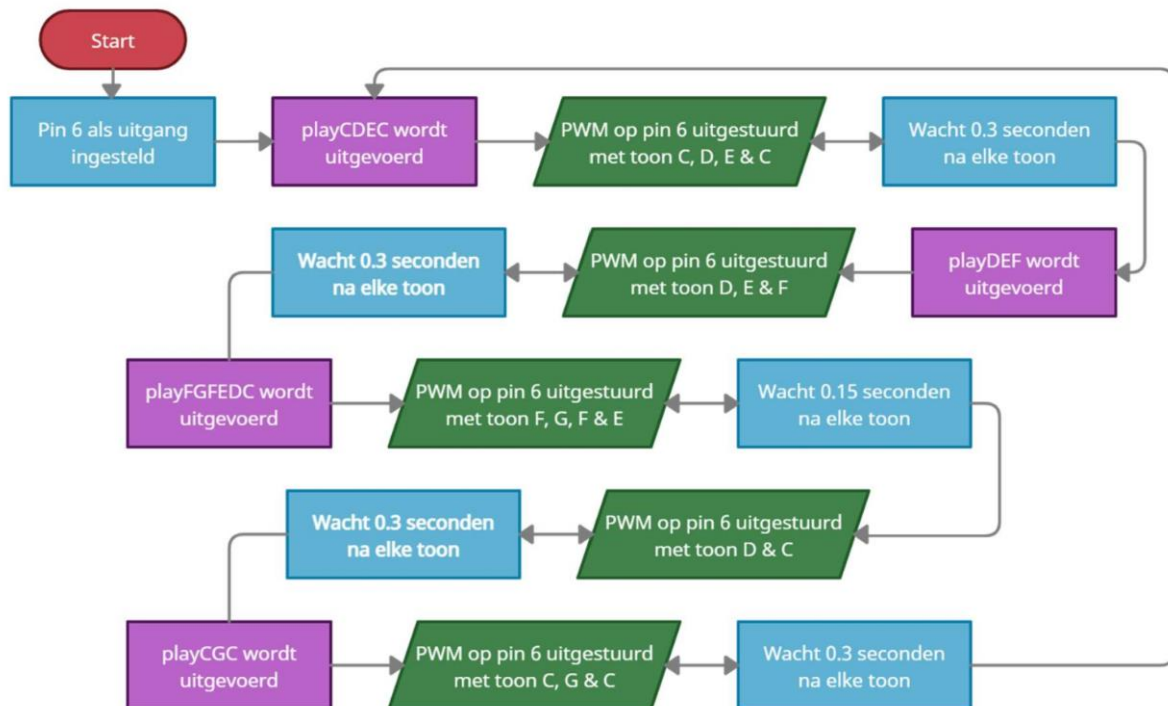
CDEC CDEC DEF DEF FGFED C FGFED C C G(392 Hz) C C G(392 Hz) C

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

Noot	n	f_n	x
A	0	440,0	0,000
Ais	1	466,2	0,056
B	2	493,9	0,109
C	3	523,3	0,159
Cis	4	554,4	0,206
D	5	587,3	0,251
Dis	6	622,3	0,293
E	7	659,3	0,333
F	8	698,5	0,370
Fis	9	740,0	0,405
G	10	784,0	0,439
Gis	11	830,6	0,470
A1	12	880,0	0,500
A2	24	1760,0	0,750

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	<ol style="list-style-type: none"> 1. De pin 6 wordt als een uitgangspijn ingesteld. 2. De tonen CDEC worden afgespeeld via de tone()-functie. 3. De tonen DEF worden afgespeeld via de tone()-functie. 4. De tonen FGFEDC worden afgespeeld via de tone()-functie. 5. De tonen CGC worden afgespeeld via de tone()-functie. 6. De code wordt terug afgespeeld vanaf punt 2. <p>Delays worden doorheen de play functies gebruikt om de beste versie van Broeder Jacob te krijgen</p>

CODE!	Plaats hier je geschreven code
	<pre> int luidspreker = 9; void setup() { Serial.begin(9600); pinMode(luidspreker, OUTPUT); } void loop() { playCDEC(); playCDEC(); playDEF(); playDEF(); playFGFEDC(); playFGFEDC(); playCGC(); playCGC(); } void playCDEC(){ playC(); </pre>

```

delay(300);
playD();
delay(300);
playE();
delay(300);
playC();
delay(300);
}
void playDEF(){
playD();
delay(300);
playE();
delay(300);
playF();
delay(600);
}
void playFGFEDC(){
playF();
delay(150);
playG();
delay(150);
playF();
delay(150);
playE();
delay(150);
playD();
delay(300);
}

```

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

 Neem een foto van het in- en uitgangssignaal weergegeven op een scope



	Ch1 (ingang)	Ch2 (uitgang)
$U_{in(max)}$	3,38 V	3,63 V
frequentie	662,3 Hz	666,7 Hz

4 Opgave 4 : creatie van sinusgolf, blokgolf en zaagtand

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


Fast_PWM_sinus §

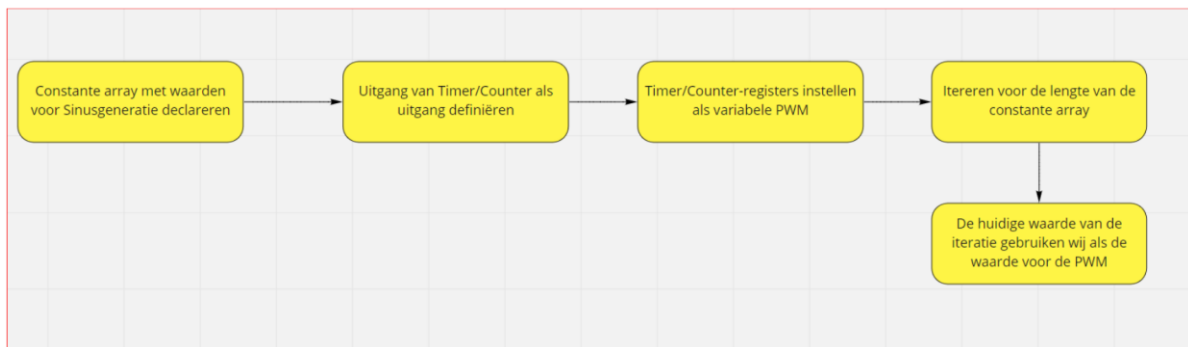
```

1 // fast PWM sinus
2 const int audioPin = 9; // uitgangspen voor audio
3 const byte value[] = {
4   128, 131, 134, 137, 141, 144, 147, 150, 153, 156, 159, 162, 168, 171,
5   174, 177, 180, 183, 186, 189, 191, 194, 197, 199, 202, 205, 207, 209, 212,
6   214, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235, 236, 238, 240, 241,
7   243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 253, 254, 254, 255,
8   255, 255, 255, 255, 255, 255, 255, 255, 254, 254, 254, 253, 253, 252,
9   251, 250, 249, 248, 247, 246, 245, 243, 242, 240, 239, 237, 236, 234, 232,
10  230, 228, 226, 224, 222, 220, 218, 215, 213, 211, 208, 206, 203, 201, 198,
11  195, 193, 190, 187, 184, 181, 179, 176, 173, 170, 167, 164, 161, 158, 155,
12  152, 148, 145, 142, 139, 136, 133, 130, 126, 123, 120, 117, 114, 111, 108,
13  104, 101, 98, 95, 92, 89, 86, 83, 80, 77, 75, 72, 69, 66, 63, 61, 58, 55,
14  53, 50, 48, 45, 43, 41, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 19, 17,
15  16, 14, 13, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 2, 1, 1, 0, 0, 0, 0, 0, 0,
16  0, 1, 1, 1, 2, 2, 3, 4, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15, 16, 18, 20, 21,
17  23, 25, 27, 29, 31, 33, 35, 37, 39, 42, 44, 47, 49, 51, 54, 57, 59, 62,
18  65, 67, 70, 73, 76, 79, 82, 85, 91, 94, 97, 101, 103, 106, 109, 112,
19  115, 119, 122, 125
20 };
21
22 void setup() {
23   pinMode(audioPin, OUTPUT);
24   TCCR1A = 0b10000001;
25   TCCR1B = 0b00001001;
26 }
27
28 void loop() {
29   //sinus
30   for (unsigned int j=0 ; j<256; j++) {
31     analogOut(value[j]);
32     delayMicroseconds(10);
33   }
34   delay(1000);
35   //blok
36   for (unsigned int b=0; b<3; b++) {
37     for (unsigned int j=0 ; j<100; j++){
38       analogOut(0);
39       delayMicroseconds(3000);
40       analogOut(255);
41       delayMicroseconds(3000);
42     }
43   }
44   delay(1000);
45   //zaag
46   for (unsigned int z=0; z<500; z++) {
47     for (unsigned int j=0; j<255 ; j++) {
48       analogOut(j);
49       delayMicroseconds(3000);
50       analogOut(255);
51       delayMicroseconds(10);
52     }
53   }
54   delay(1000);
55 }
56
57 void analogOut(byte val) {
58   OCR1A = (val);
59 }

```

4.1 De sinusgolf

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



Vraag1 (7	Verklaar het genereren van het sinussignaal via gebruik van timer 1 en PWM (waarvoor dienen TCCR1A, TCCR1B, OCRA en de functie analogOut in bovenstaand programma?)
Antwoord	<p>Via de configuratie van de registers van Timer/Counter1 (TCCR1A en TCCR1B) wordt “non-inverting Fast-PWM”-modus ingesteld met een 8-bits resolutie en een klok gelijk aan 16 MHz.</p> <p>In deze modus zal de Timer/Counter1 optellen tot zijn maximale waarde en bij een “Compare Match (waarde OCRA = Telwaarde Counter)” zal de uitgang laag worden. De functie “analogOut” kopieert de huidige waarde van de iteratie in de array naar het OCRA-register</p>

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


CODE!	Plaats hier je geschreven code
	<pre> 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, 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, 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() { // put your setup code here, to run once: pinMode(audioPin, OUTPUT); </pre>

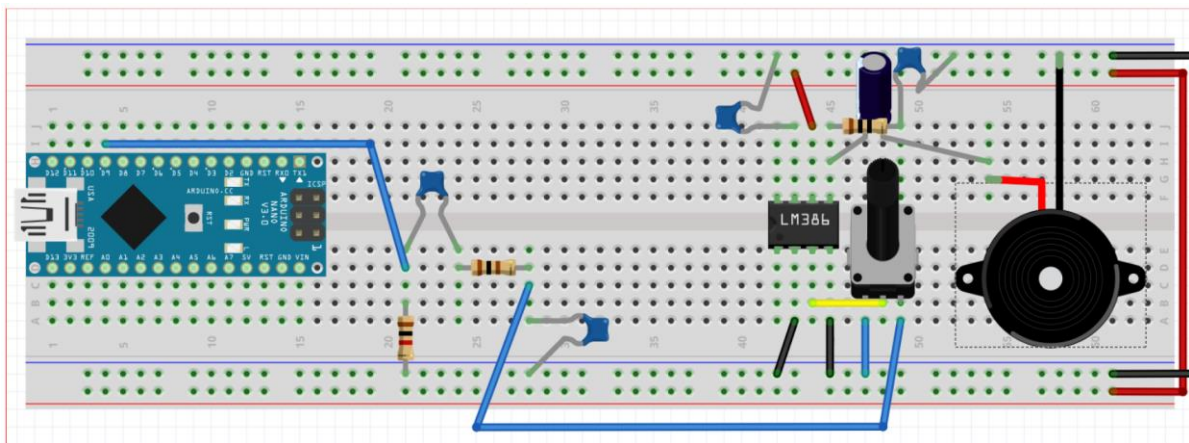
```

TCCR1A = 0b10000001;
TCCR1B = 0b00001001;
}
void loop() {
// put your main code here, to run repeatedly:
//sinus
for (unsigned int j = 0; j < 256; j++) {
analogOut(value[j]);
delayMicroseconds(40);
}
}
void analogOut(byte val) {
OCR1A = (val);
//delayMicroseconds(5);
}

```

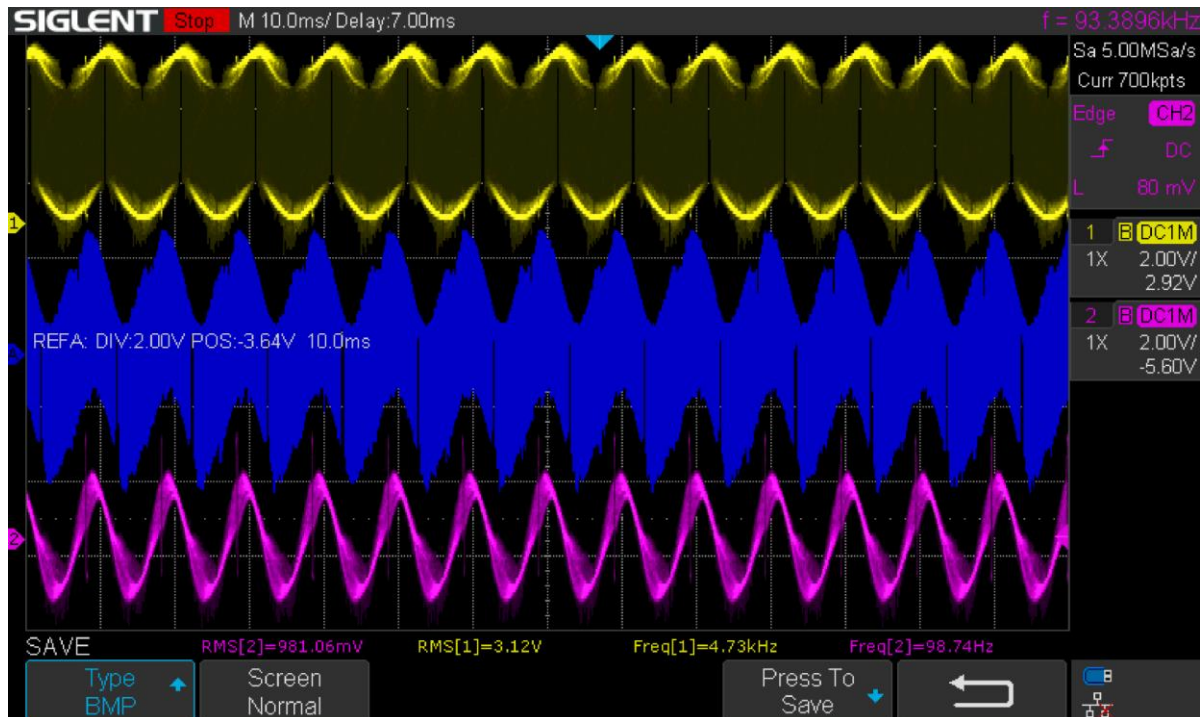
Teken in Fritzing (<http://fritzing.org/home/>) de aansluitingen van Arduino met het RC-filter (filterschakeling voor de versterker (zie opgave 1))

 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.



Geel => CH1 | U_{R1} => Blauw | U_{C2} => Paars

	Ch1 (ingang)	Ch2 (U_{R1})	Ch3 (U_{C2})
$U_{in(max)}$	4,4 V	4,4 V	1 V
frequentie	62 kHz	98 Hz	100 Hz

Vraag 8	Verklaar de werking van de hardwareschakeling aangaande het omvormen van het PWM-signaal naar een sinusvorm
Antwoord	De uitgang van de Timer/Counter1 levert een variabel PWM-signaal welke overeenkomt met de opgeslagen waarde in het OCR1A-register. De duty-cycle van het PWM-signaal is evenredig met een DC-component. Het PWM-signaal wordt door een bandfilter (75 Hz – 15 kHz) gestuurd waardoor de oorspronkelijke PWM-frequentie wordt gedempd en enkel de variërende “DC-component” overblijft. Het filter op zich is niet perfect (onvoldoende demping bij de PWM-frequentie waardoor nog een zker HF-component gedeelte overblijft).

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

CODE!	Plaats hier je aangepaste code
	<pre> void loop() { //sinus for (unsigned int j = 0; j < 256; j++) { analogOut(value[j]); //delayMicroseconds(40); } } </pre>

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

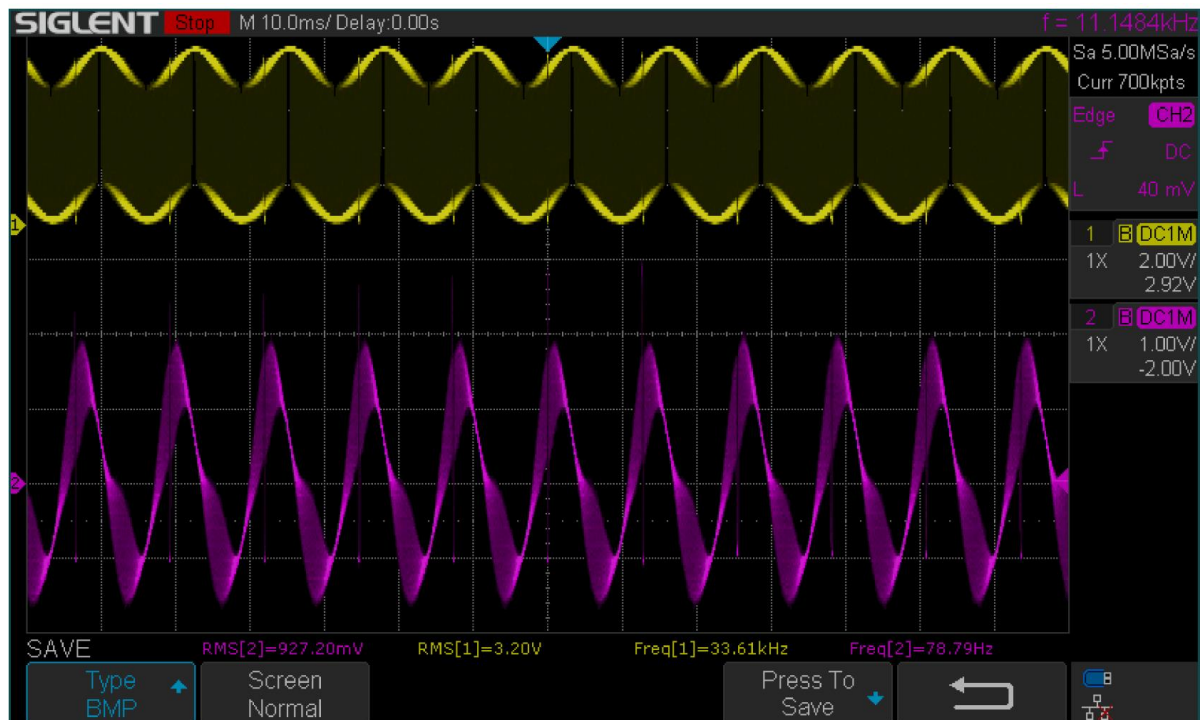


Ch1 (U_{C2})	
$U_{in(max)}$	1,8 V
frequentie	5,65 kHz

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

CODE!	Plaats hier je aangepaste code
	<pre>void loop() { //sinus for (unsigned int j = 0; j < 256; j++) { analogOut(value[j]); delayMicroseconds(50); } }</pre>

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



Ch1 (U_{C2})	
$U_{in(max)}$	1,88 V
frequentie	78,8

Hz


Bijzondere klanken: kerkklok

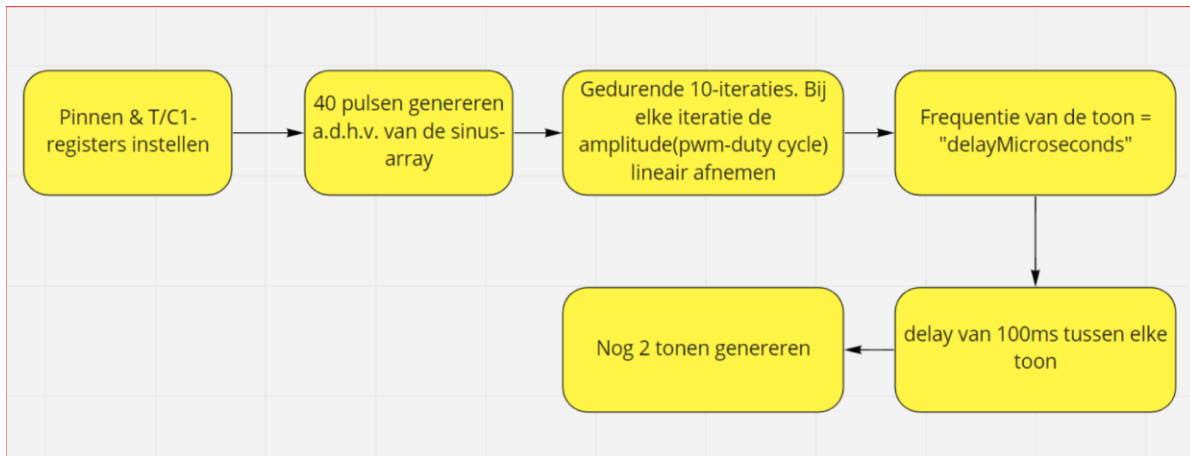
Voorbeeldcode om een kerkklok-sound te bekomen:

```

1 // fast PWM kerkklok
2
3 const int audioPin = 9; // uitgangspen voor audio
4 const byte value[] = {
5     128, 131, 134, 137, 141, 144, 147, 150, 153, 156, 159, 162, 168, 171,
6     174, 177, 180, 183, 186, 189, 191, 194, 197, 199, 202, 205, 207, 209, 212,
7     214, 217, 219, 221, 223, 225, 227, 229, 231, 233, 235, 236, 238, 240, 241,
8     243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 253, 254, 254, 255,
9     255, 255, 255, 255, 255, 255, 255, 255, 255, 254, 254, 254, 253, 253, 252,
10    251, 250, 249, 248, 247, 246, 245, 243, 242, 240, 239, 237, 236, 234, 232,
11    230, 228, 226, 224, 222, 220, 218, 215, 213, 211, 208, 206, 203, 201, 198,
12    195, 193, 190, 187, 184, 181, 179, 176, 173, 170, 167, 164, 161, 158, 155,
13    152, 148, 145, 142, 139, 136, 133, 130, 126, 123, 120, 117, 114, 111, 108,
14    104, 101, 98, 95, 92, 89, 86, 83, 80, 77, 75, 72, 69, 66, 63, 61, 58, 55,
15    53, 50, 48, 45, 43, 41, 38, 36, 34, 32, 30, 28, 26, 24, 22, 20, 19, 17,
16    16, 14, 13, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 2, 1, 1, 0, 0, 0, 0, 0, 0,
17    0, 1, 1, 1, 2, 2, 3, 4, 4, 5, 6, 7, 8, 10, 11, 12, 13, 15, 16, 18, 20, 21,
18    23, 25, 27, 29, 31, 33, 35, 37, 39, 42, 44, 47, 49, 51, 54, 57, 59, 62,
19    65, 67, 70, 73, 76, 79, 82, 85, 91, 94, 97, 101, 103, 106, 109, 112,
20    115, 119, 122, 125
21 };
22
23 void setup() {
24     pinMode(audioPin, OUTPUT);
25     TCCR1A = 0b10000001;
26     TCCR1B = 0b00001001;
27 }
28
29 void loop() {
30
31     for (int ampl = 10; ampl >=0; ampl--) {
32         for (unsigned int d = 0; d<40; d++) {
33             for (unsigned int j = 0; j<256; j++) {
34                 analogOut(value[j] /10 * ampl);
35                 delayMicroseconds(5);
36             }
37         }
38     }
39
40     delay(100);
41
42     for (int ampl = 10; ampl >=0; ampl--) {
43         for (unsigned int d = 0; d<40; d++) {
44             for (unsigned int j = 0; j<256; j++) {
45                 analogOut(value[j] /10 * ampl);
46                 delayMicroseconds(5);
47             }
48         }
49     }
50
51     delay(100);
52
53     for (int ampl = 10; ampl >=0; ampl--) {
54         for (unsigned int d = 0; d<20; d++) {
55             for (unsigned int j = 0; j<256; j++) {
56                 analogOut(value[j] /10 * ampl);
57                 delayMicroseconds(10);
58             }
59         }
60     }
61 }
62
63 void analogOut(byte val) {
64     OCR1A = (val);
65 }
66

```

 **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	De code genereert twee keer een toon van 784,3 Hz en één keer een toon van 392 Hz. Beide tonen bestaan uit 10 keer 40 pulsen en bij elke iteratie van de “ampl” for-loop neemt de amplitude van de toon met 1 af.

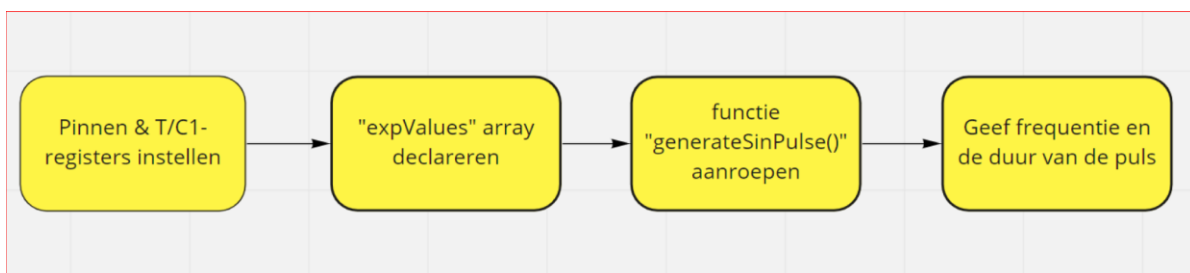
Voer de code uit.

Vraag 10	Wat is jouw indruk van het geluid?
Antwoord	Door het feit dat het geluid niet exponentieel daalt maar lineair klinkt het eerder robotachtig.

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
<pre> 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, </pre>	

```

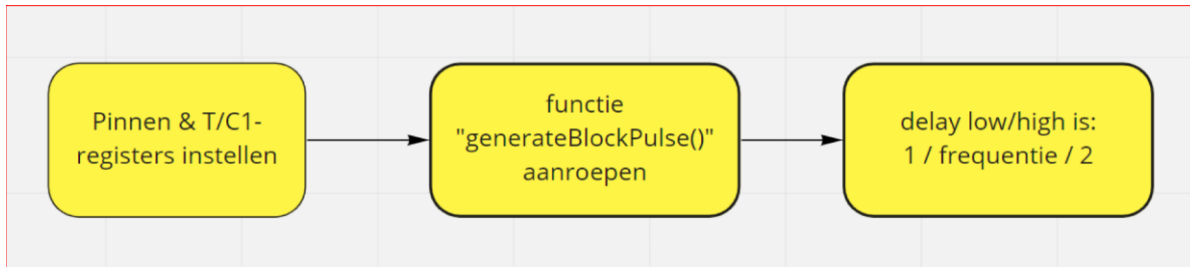
255, 255, 255, 255, 255, 255, 255, 255, 254, 254, 254, 253, 253, 252, 251,
250, 249, 248, 247, 246, 245, 243, 242, 240, 239, 237, 236, 234, 232, 230,
228, 226, 224, 222, 220, 218, 215, 213, 211, 208, 206, 203, 201, 198, 195,
193, 190, 187, 184, 181, 179, 176, 173, 170, 167, 164, 161, 158, 155, 152,
148, 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,
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
};
const double expValues[] = {0, 0.5, 1, 1.5, 2, 2.2, 3, 5.5, 5, 10};
void setup() {
  pinMode(audioPin, OUTPUT);
  TCCR1A = 0b10000001;
  TCCR1B = 0b00001001;
}
void loop() {
  generateSinPulse(90, 5);
  delay(5);
  generateSinPulse(90, 5);
  delay(10);
  generateSinPulse(180, 5);
  delay(20);
  generateSinPulse(90, 5);
  delay(5);
  generateSinPulse(90, 5);
  delay(10);
  generateSinPulse(180, 5);
  delay(1000);
}
void generateSinPulse(int freq, int duration) {
  int freqToMicro = (1 / freq) / 255;
  for (int ampl = 9; ampl >= 0; ampl--) {
    for (unsigned int d = 0; d < duration; d++) {
      for (unsigned int j = 0; j < 256; j++) {
        analogOut(value[j] / 10 * expValues[ampl]);
        delayMicroseconds(freqToMicro);
      }
    }
  }
}
void analogOut(byte val) {
  OCR1A = (val);
  //delayMicroseconds(5);
}

```

4.2 De blok golf

Schrijf een sketch om een blok golf 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 blok golf met een bepaalde frequentie kan genereren vanuit Arduino.



Vraag 11	Verklaar hoe je een blok golf van 100 Hz kan genereren aan de hand van je getekende stroomdiagram
Antwoord	<p>Voor de codegeneratie is de generatie van de blok golf zelf hier in een functie gestoken. Op deze wijze kan de blok golf worden opgeroepen met frequentie en het aantal pulsen als argumenten.</p> <p>De frequentie wordt bepaald door de 2 “delayMicroseconds”. De totale delay komt overeen met de periode van de gewenste frequentie : Delay = $(1/\text{frequentie})/2$</p>

CODE!	Plaats hier je aangepaste code
	<pre> 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, 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; } void loop() { //blok generateBlockPulse(100, 200); delay(1000); </pre>

```

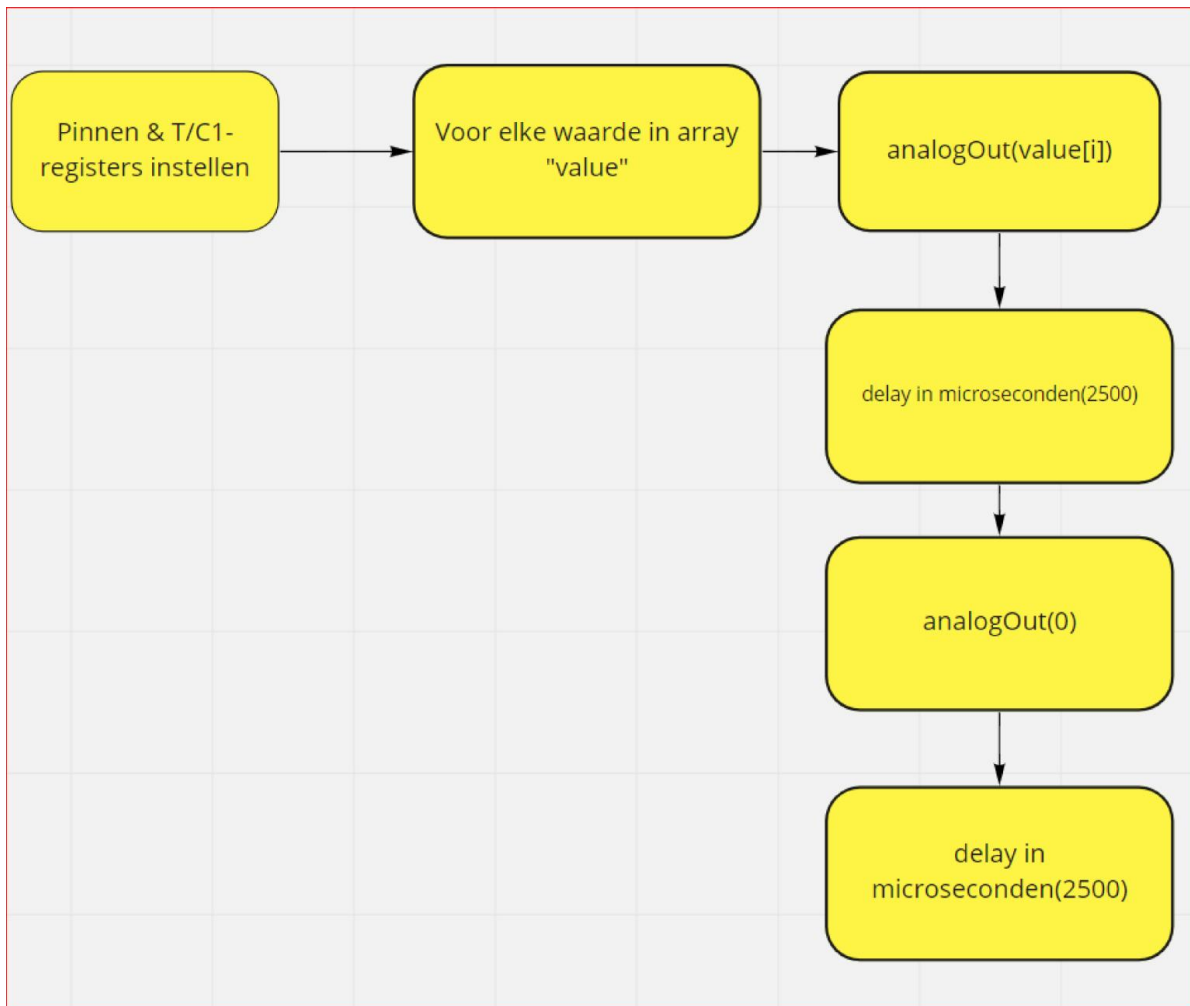
}
void generateBlockPulse(float freq, int numPulses) {
float freqToDelay = (1/freq) / 2 * 1000000;
for (unsigned int j = 0; j < numPulses; j++) {
analogOut(0);
delayMicroseconds(freqToDelay);
analogOut(255);
delayMicroseconds(freqToDelay);
}
}
void analogOut(byte val) {
OCR1A = (val);
//delayMicroseconds(5);
}

```

4.3 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	De curve van een zaagtand wordt geïmiteerd door de DC-spanning lineair te laten stijgen. Bij maximale waarde (255) wordt de PWM-duty-cycle terug naar 0 gebracht zodat de DC-spanning 0 V wordt.

CODE!	Plaats hier je aangepaste code
	<pre> const int audioPin = 9; //uitgangspen voor audio const byte value[] = { 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, </pre>

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
	<pre> 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255 }; void setup() { pinMode(audioPin, OUTPUT); TCCR1A = 0b10000001; TCCR1B = 0b00001001; } void loop() { //zaagtand for (int z = 0; z < 100; z++) { for (unsigned int j = 128; j > 0; j--) { analogOut(value[j]); delayMicroseconds(2500); analogOut(255); delayMicroseconds(2500); } analogOut(0); delayMicroseconds(5000); } } void analogOut(byte val) { OCR1A = (val); //delayMicroseconds(5); } </pre>

Vraag 13	Wat is jouw indruk van het geluid?
Antwoord	Het geluid geeft de indruk van een snaar die aangeslagen wordt

5 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. Bijvoorbeeld : 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 display (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

Code van de sketch:

```
#include <LiquidCrystal_I2C.h>
```

```
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, 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
};
```

```
#define pot A3
```

```
#define sw 2
```

```

#define signalPin 9

LiquidCrystal_I2C lcd(0x27, 20, 4);

int f = 800;

void setup()
{
  Serial.begin(9600);
  pinMode(sw, INPUT);
  pinMode(pot, INPUT);
  pinMode(signalPin, OUTPUT);
  TCCR1A |= (1 << COM1A1);
  TCCR1A |= (1 << WGM10);
  TCCR1B |= (1 << CS10);
  TCCR1B |= (1 << WGM12);
  lcd.init();
  lcd.backlight();
}

void loop() {
  lcd.clear();

  f = map(analogRead(pot), 0, 1023, 100, 1000);
  lcd.setCursor(0, 0);
  lcd.print(f);
  lcd.setCursor(8, 0);
  lcd.print("Hz");

  Serial.print(f);
  Serial.print("Hz  ");

```

```

if (digitalRead(sw)) {

    double a = 1.0 / f;
    double b = a * 1000000.0;
    double c = b / 255;
    lcd.setCursor(0, 1);
    lcd.print("sinus");
    Serial.println("sinus");
    //39 -> 3.9
    //sinus
    for (int j = 0; j < 100; j++)
    {
        for (int j = 0; j < 256; j++)
        {
            analogOut(value[j]);
            delayMicroseconds(c);
        }
    }
}

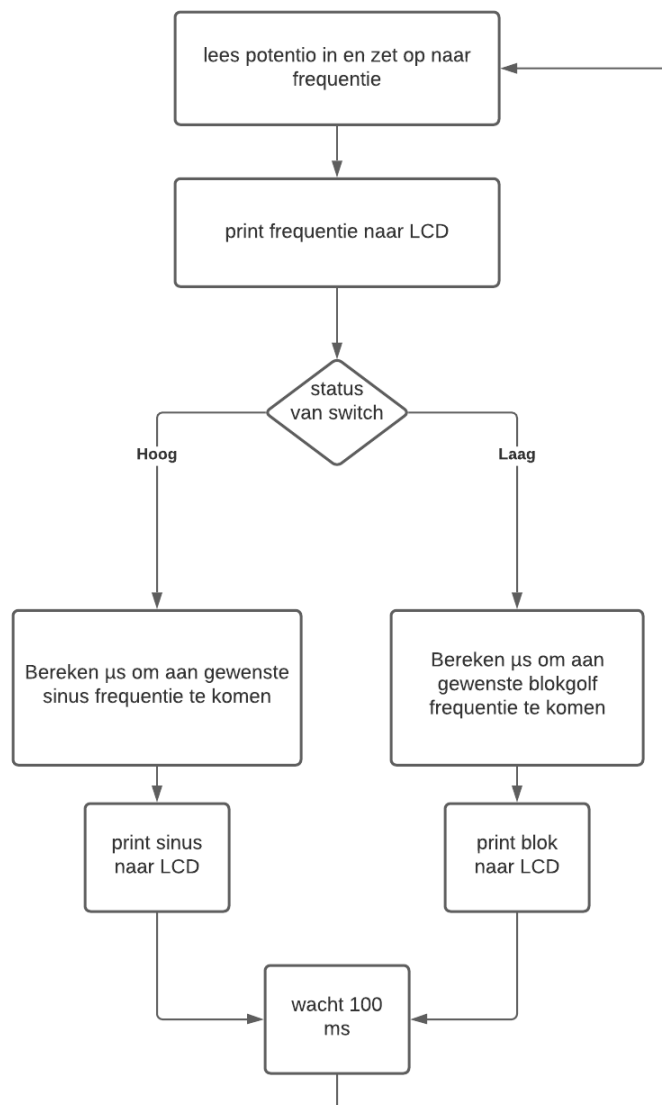
if (!digitalRead(sw)) {
    double a = 1.0 / f;
    double b = a * 1000000.0;
    double c = b / 2.0;
    lcd.setCursor(0, 1);
    lcd.print("blok");
    Serial.println("blok");

    for (int j = 0; j < 3; j++)
    {
        for (int x = 0; x < 100; x++)

```

```
{  
    analogOut(0);  
    delayMicroseconds(c);  
    analogOut(255);  
    delayMicroseconds(c);  
}  
}  
delay(100);  
}  
}  
  
void analogOut(byte val) {  
    OCR1A = (val);  
}
```

Stroomdiagram van de werking van de schakeling



Aansluitschema

