3 – Geluid via Arduino

Ing. Patrick Van Houtven

ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

Geluiden via Arduino

Welk soort geluid?

- Naast het geven van optische uitvoer (LED – display) zijn microcontrollers ook geschikt om akoestische signalen te genereren. (Denk aan de biebjes van een computer)
- Eenvoudige geluiden (16 Hz – 16 kHz) zijn met een µC (microcontroller) gemakkelijk te realiseren.
- De geluiden die opgewekt kunnen worden variëren van eenvoudige orgelgeluiden tot verfijnde synthesizergeluiden.
- Er zijn ook meer ingewikkelde geluiden realiseerbaar met een µC. Hoewel de resultaten op zich nog niet helemaal bevredigend zijn is ook spraaksynthese mogelijk met een arduino.

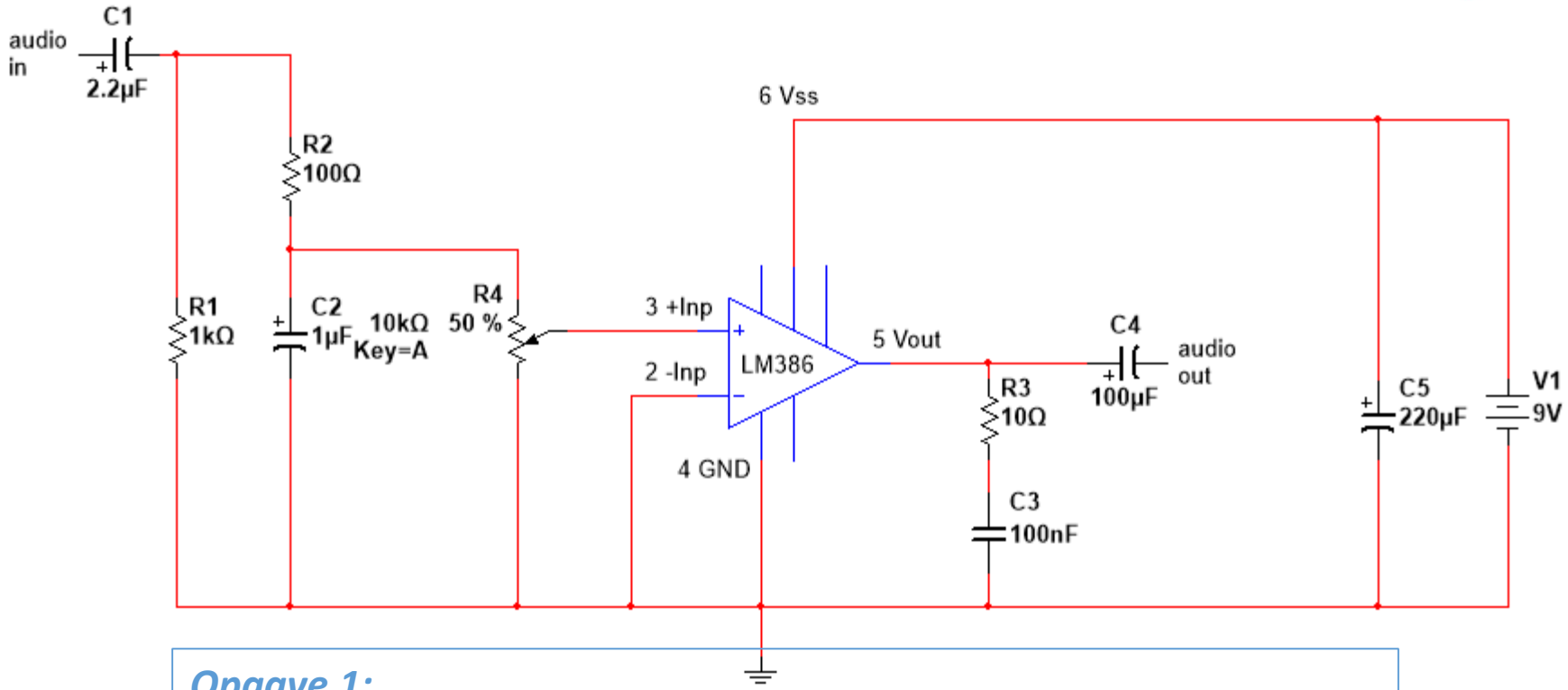
Wat is belangrijk?

- Geluid opwekken met tone()-functie
- Geluid opwekken met PWM
- Principe omschrijven hoe de PWM-mode werkt van Timer/Counter1
- Werkingsprincipe capacitieve sensor omschrijven

3-1 De tone()-functie

Versterken van een toon

- Voorbeeldschakeling met LM386.
- Aan audio-out kan een luidspreker gehangen worden met een impedantie van 8 Ohm of hoger. Indien de luidspreker de schakeling te zwaar belast kan in serie met de luidspreker een weerstand geplaatst worden van enkele honderden ohm. Bijvoorbeeld 370 Ohm.
- Audio-in wordt verbonden met een OUTPUT-pen van de Arduino.
- De condensator van 1µF zorgt voor een bepaalde LD-filtering. Dit is vooral handig wanneer audiotonen met behulp van PWM worden gegenereerd. De hogere PWM-frequenties worden dan onderdrukt terwijl de gewenste lagerfrequent audiotonen de ingang van de versterker onverzwakt bereiken.



Opgave 1:
Bouw de versterker en gebruik deze om de tonen te kunnen afspelen van de volgende opgaven in deze presentatie.

3-1 De tone()-functie

tone()

- Genereert een blokgolf met de gespecificeerde frequentie en met een duty cycle gelijk aan 50%
- Een duurtijd van de toon kan gespecificeerd worden. Indien geen specifieke duurtijd wordt opgegeven duurt de toon tot de functie noTone() wordt opgeroepen.
- Aan de betreffende pin waar de toon wordt gegenereerd kan een piëzo buzzer of luidspreker worden aangesloten.
- Er kan maar één toon terzelfdertijd gegenereerd worden. Indien op een bepaalde pin reeds een toon wordt afgespeeld zal het oproepen van tone() op een andere pin geen enkel effect hebben. Indien de toon op dezelfde pin wordt afgespeeld, zal met het oproepen van tone() de frequentie gezet worden. Indien je meerdere tonen wil weergeven over meerdere pinnen moet je eerst noTone() voor een bepaalde pin oproepen alvorens een andere pin via tone() van een bepaalde toon wordt voorzien.
- Het gebruik van de tone()-functie zal interfereren met de PWM-output op de pinnen 3 en 11 (op alle borden behalve Mega)
- Welke frequenties?

Board	Min frequentie (Hz)	Max frequentie (Hz)
Uno, Mega, Leonardo en andere AVR-borden	31	65535
Gemma	Niet geïmplementeerd	Niet geïmplementeerd
Zero	41	27500
Due	Niet geïmplementeerd	Niet geïmplementeerd

3-1 De tone()-functie

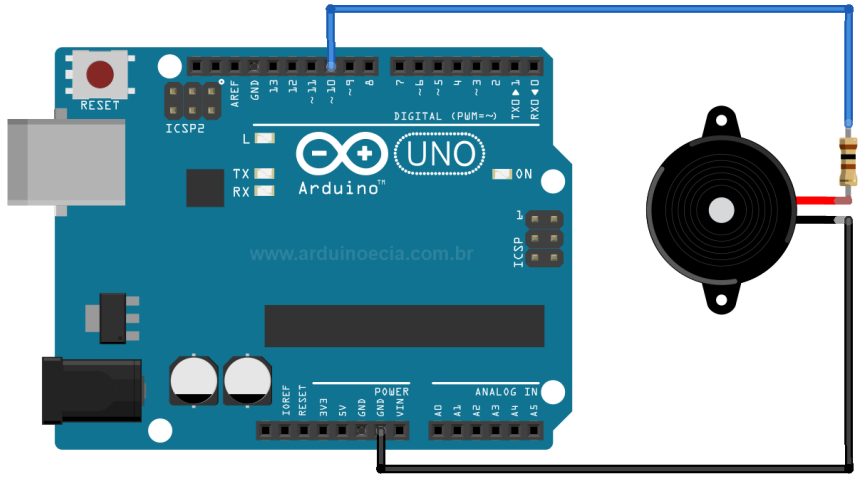
Syntax tone()

- tone(pin, frequency)
- tone(pin, frequency, duration)

Parameters

- pin de arduino-pen waarmee de toon wordt gegenereerd
- frequency : de frequentie van de toon in Hz (unsigned int)
- duration : de lengte van de toon in milliseconde (optioneel – unsigned long)

Return-waarde : niets



Syntax noTone()

- noTone(pin)

Parameter

- pin de arduino-pen waarop de lopende toon wordt gegenereerd

Return-waarde : niets

3-1 De tone()-functie

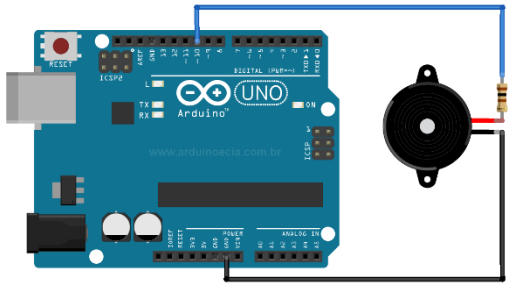
Eenvoudig intermitterend akoestisch alarm

```
// voorbeeld van een eenvoudig akoestisch alarm

// instellen variabele luidspreker voor gebruik poort 10
const int luidspreker = 10;

void setup() {
  // instellen pin voor luidspreker (pen 10) als output
  pinMode(luidspreker, OUTPUT);
}

void loop() {
 // afspelen van toon met f = 523.3 Hz en duurtijd 450 ms
  tone(luidspreker, 523.3, 450);
 // instellen vertragingstijd van 1000 ms
  delay(1000);
}
```



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

Tabel met frequenties muzieknoten

Opgave 2:
Voer de sketch uit en gebruik de LM386-versterker om de luidspreker aan te sluiten op arduino.
Pas de sketch aan zodat de µC 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
Pas de duur van de tonen aan het liedje aan en voeg wachttijden in daar waar nodig (artistieke vrijheid is toegestaan ☺)
Teken het stroomdiagram van de sketch.

3-1 De tone()-functie

Voorbeeld van groter alarm (sirene)

```
// voorbeeld van een groot alarm

// instellen variabele luidspreker voor gebruik poort 10
const int luidspreker = 10;

void setup() {
  // instellen pin voor luidspreker (pen 9) als output
  pinMode(luidspreker, OUTPUT);
}

void loop() {
  // verhogen van de toon (sirene-vorm)
  for (int i=200; i<500; i += 10) {
  tone(luidspreker, i, 50);
  delay(20);
  }

 //verlagen van de toon (sirene-vorm)
  for (int j=200; j<500; j += 1) {
  tone(luidspreker, (550-j), 50);
  delay(20);
  }

}
```

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

Opgave3 :
Pas de 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 het stroomdiagram weer van de sketch.

3-2 Geluiden met PWM

- (fast) PWM is de eenvoudigste manier om snel werkelijke geluiden te maken.
- Verschil toon – geluidsgolf
  - Toon kan volledig worden beschreven met één parameter, de frequentie
  - De toon van “A” bij een muziekinstrument is 440 Hz. Toch klinkt bijvoorbeeld een piano anders dan een gitaar bij het aanslaan van deze “A”. De reden is dat geluid veel meer eigenschappen bevat dan de toon alleen. Eigenschappen zoals timbre (bepaald door mix van de boventonen) en amplitude (verandering van de omhullende in de tijd) zijn van groot belang om geluid te specificeren.
  - De tone()-functie kan de timbre en amplitude niet beïnvloeden. Vermits blokgolven rijk aan boventonen zijn, klinken ze ‘scherp’ en ‘robotachtig’. Voor minder scherpe tonen moeten deze boventonen onderdrukt worden. Hiervoor hebben we sinusvormige golven nodig.
  - Zuivere sinusvormige signalen zijn niet te realiseren door een poortpen aan en uit te schakelen (zoals tone() doet). Een mogelijkheid om ze te creëren is gebruik te maken van PWM.
    - Principe : frequentie gebruiken (bv. 62,5 kHz) die ver boven de hoogste audio-frequentie ligt. Deze frequentie blijft constant maar door de pulsbreedte te variëren kunnen geluidsgolven opgewekt worden.
    - Met een passend LD-filter kan dan vrijwel elke willekeurige golfvorm benadert worden.

Geluiden met PWM

Opwekken sinusvorm

```
// 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
 };
```

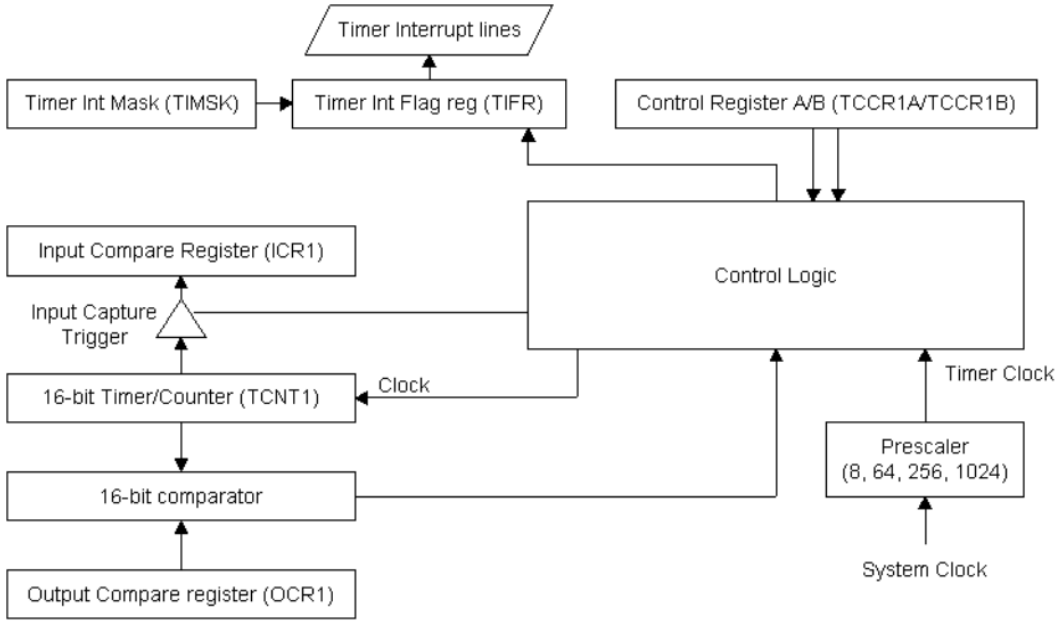
```
void setup() {
  pinMode(audioPin, OUTPUT);
  TCCR1A = 0b10000001;
  TCCR1B = 0b00001001;
 }

void loop() {
  for (unsigned int j=0; j<256; j++) {
    analogOut(value[j]);
    delayMicroseconds(10);
  }
}
void analogOut(byte val) {
  OCR1A = (val);
}
```

Sinus bepaald en opgeslagen in een array genaamd value[].
Hierdoor moeten de sinuswaarden niet meer berekend worden en kan in het hoofdprogramma de afzonderlijke waarden via fast PWM naar buiten gevoerd worden.
Omwille van de snelheid zijn de sinuswaarden op voorhand berekend.

TCCCR1A en TCCCR1B

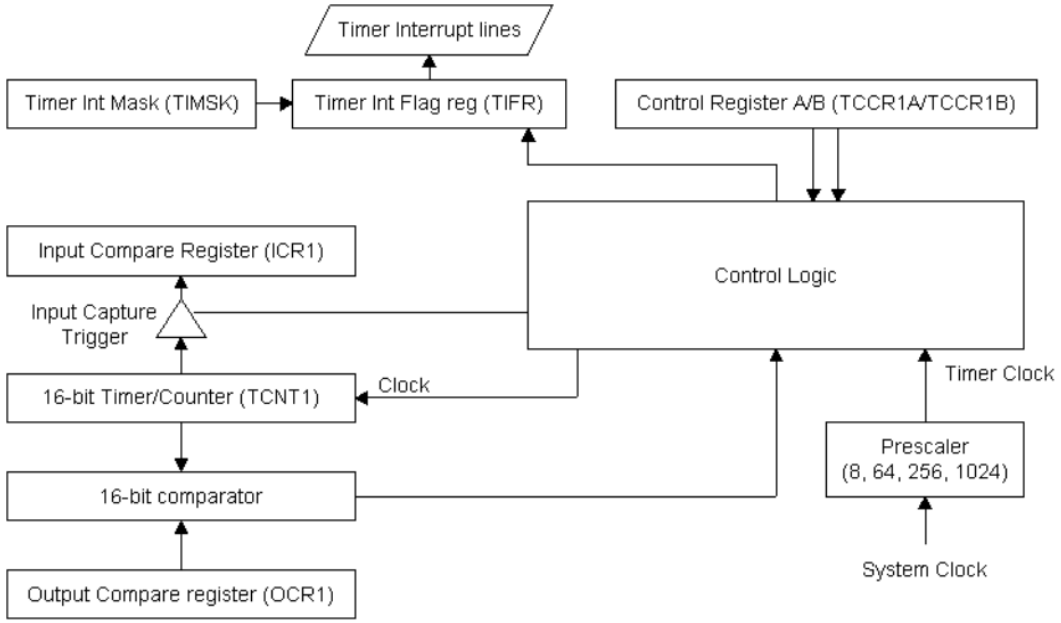
- Timers tellen normaal klokcycli. De klokcycli kunnen afkomstig zijn van de systeemklok of vertraagd worden door gebruik te maken van de prescaler (delen door 8, 64, 256 of 1024)
- Een 16-bit Timer heeft een bereik van 0xFFFF = 65536 telwaarden.
  - Stel dat de prescaler ingesteld staat op 1024 => 65536 x 1024 klokcycli die geteld kunnen worden of 67108864 klokcycli vooraleer een overflow optreedt.
  - Stel dat de klok van het bordje ingesteld staat op 8 MHz. 1 klokcyclus duurt dan 1/8MHz of 0,125µs.
  - De timer genereert dan een overflow om de 67108864 klokcycli x 0,125µs of om de 8,88608 seconden.



3-2 Geluiden met PWM

- TCCR1A (TimerCounterCaptureCompareRegister1A)

  [TRCCR1A = 0b10000001]



TCCR1A:

Bit 7							Bit 0
COM1A1	COM1A0	---	---	---	---	PWM11	PWM10

Here's what the individual bits do:

COM1A1/COM1A0: Compare Output Mode bits 1/0; These bits control if and how the Compare Output pin is connected to Timer1.

COM1A1	COM1A0	Compare Output Mode
0	0	Disconnect Pin OC1 from Timer/Counter 1
0	1	Toggle OC1 on compare match
1	0	Clear OC1 on compare match
1	1	Set OC1 on compare match

With these bit you can connect the OC1 Pin to the Timer and generate pulses based on the timer. It's further described below.

PWM11/PWM10: Pulse Width Modulator select bits; These bits select if Timer1 is a PWM and it's resolution from 8 to 10 bits:

PWM11	PWM10	PWM Mode
0	0	PWM operation disabled
0	1	Timer/Counter 1 is an 8-bit PWM
1	0	Timer/Counter 1 is a 9-bit PWM
1	1	Timer/Counter 1 is a 10-bit PWM

- TCCR1B (TimerCounterCaptureCompareRegister1B) [TRCCR1B = 0b00001001]

- ICNC1
  - Input Capture Noise Canceler
  - ICNC1=1 : Noise Canceler op pin ICP (Input Capture Pin) is geactiveerd en triggert de input capture na 4 gelijke samples. De flank waarop getriggerd wordt, wordt bepaald door ICES1.
- ICES1
  - Input Capture Edge Select
  - ICSE1 = 0 : de inhoud van TCN1 (Timer/Counter Register1) wordt doorgegeven op de stijgende flank van de ICP pin.
- CTC1 : Clear Timer/Counter1 on Compare Match
  - CTC1 = 1 : TNCT1 register is cleared bij compare match.
  - Deze bit kan gebruikt worden om herhalende interrupts te creëren voor bv. dender te vermijden bij drukknoppen of andere op frequentie gebasseerde gebeurtenissen.
  - Timer 1 wordt hierdoor ook gebruikt in normale mode. Vergeet deze bit niet te wissen wanneer de compare match mode wordt verlaten. Anders zal de timer nooit een overflow geven en is de timing foutief.

TCCR1B:

Bit 7							Bit 0
ICNC1	ICES1	---	---	CTC1	CS12	CS11	CS10

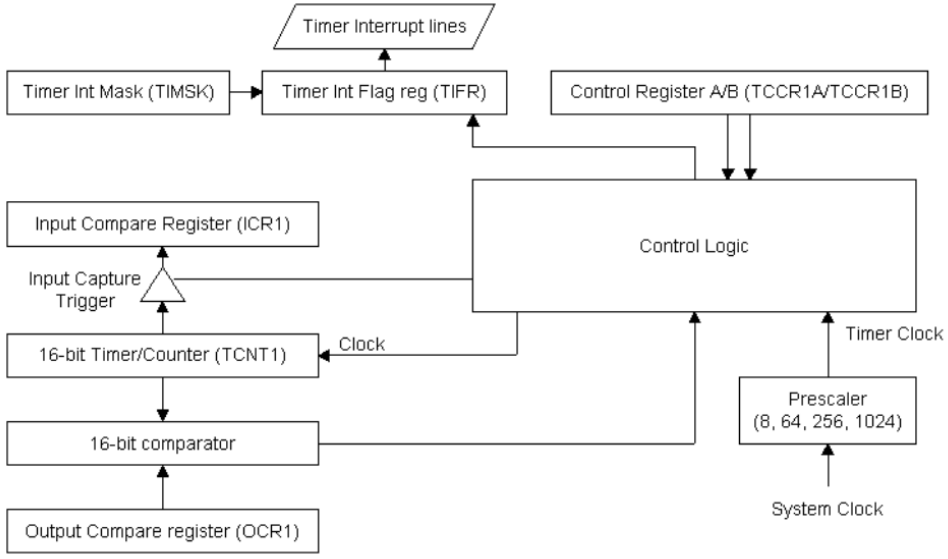
CS12	CS11	CS10	Mode Description
0	0	0	Stop Timer/Counter 1
0	0	1	No Prescaler (Timer Clock = System Clock)
0	1	0	divide clock by 8
0	1	1	divide clock by 64
1	0	0	divide clock by 256
1	0	1	divide clock by 1024
1	1	0	increment timer 1 on T1 Pin falling edge
1	1	1	increment timer 1 on T1 Pin rising edge

3-2 Geluiden met PWM

TCCR1A = 0b10000001
- Timer/Counter1 = 8-bit PWM
- Clear OCR1 by match

TCCR1B = 0b00001001
- Geen prescaler in gebruik
- Clear Timer/Counter1 on compare match

OCR1A = value[j]
- Te vergelijken waarde (0-256) voor pulsbreedte PWM




```
// 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;
 }

void loop() {
  for (unsigned int j=0; j<256; j++) {
    analogOut(value[j]);
    delayMicroseconds(10);
  }
}
void analogOut(byte val) {
  OCR1A = (val);
}
```

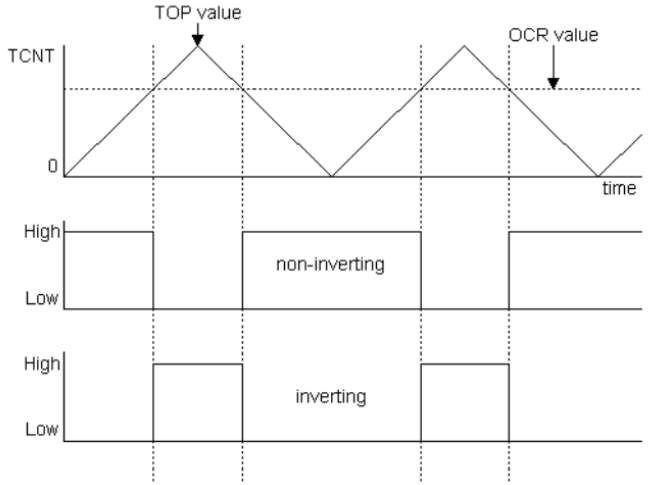
3-2 Geluiden met PWM

Principe generatie PWM

Mode PWM

COM1A1	COM1A0	Effect:
0	0	PWM disabled
0	1	PWM disabled
1	0	Non-inverting PWM
1	1	inverting PWM

PWM11	PWM10	Resolution	TOP-value	PWM Frequency
0	0	PWM function disabled		
0	1	8 bits	$00FF	fclock/510
1	0	9 bits	$01FF	fclock/1022
1	1	10 bits	$03FF	fclock/2046



Telwaarden tussen 0 – 255
Timer telt opwaarts en bij Non-inverting PWM:

- TCNT match OCR-value => OC1-pin wordt gecleared en TCNT blijft doortellen tot TOP-value (255)
- TCNT = topvalue => timer begin neerwaarts te tellen en als TCNT match OCR-value => OC1-pin terug geset
- TCNT bereikt 0-waade => terug opwaarts tellen enz…

```
// 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;
 }

void loop() {
  for (unsigned int j=0; j<256; j++) {
    analogOut(value[j]);
    delayMicroseconds(10);
  }
}
void analogOut(byte val) {
  OCR1A = (val);
}
```

opwekken blokgolf - zaagtand

```
22 void setup() {
23   pinMode(audioPin, OUTPUT);
24   TCCR1A = 0b10000001;
25   TCCR1B = 0b00001001;
26  }
27
28 void loop() {
29   //blokgolf
30   for (unsigned int b=0; b<3; b++) {
31     for (unsigned int j=0; j<100; j++) {
32     analogOut(0);
33     delayMicroseconds(3000);
34     analogOut(255);
35     delayMicroseconds(3000);
36     }
37   }
38   delay(1000);
```

```
40   // zaagtand
41   for (unsigned int z=0; z<500; z++) {
42     for (unsigned int j=0; j<255; j++) {
43     analogOut(j);
44     delayMicroseconds(3000);
45     analogOut(255);
46     delayMicroseconds(10);
47     }
48   }
49
62 }
63 void analogOut(byte val) {
64   OCR1A = (val);
65 }
```

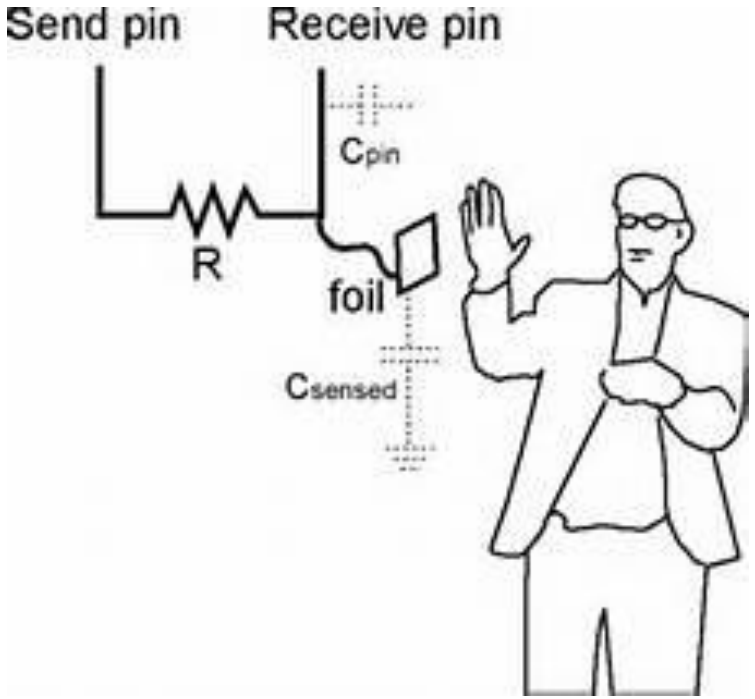
Opgave4 :
Maak sketches voor sinus, blok en zaagtand en speel deze af Maak de signaalvorm zichtbaar op de oscilloscoop en maak een schermafdruk (na eerst het beeld stilgezet te hebben)
Plaats de geluidsvormen achter elkaar en speel deze na elkaar af met 1 seconde delay tussen de vormen. Wat zijn je indrukken? Vergelijk eveneens met de tone()-functie.
Probeer een volumeaanpassing te maken door de sinuswaarde van lage amplitude naar groot te laten gaan en omgekeerd.

3-3 Capacitieve sensor

Principe capacitieve sensor

Capacitive touch sensing is een manier van menselijke aanraakdetectiemiddel dat weinig of geen kracht nodig heeft om te activeren. Het kan worden toegepast om menselijke contact te detecteren via kunststof, hout, keramiek of een ander isolatiemateriaal.

- Waarom gebruik maken van een capacitieve sensor?
  - Iedere touch sensor heeft maar 1 aansluitdraad nodig
  - Kan verborgen worden onder elk soort niet-metaalachtig materiaal
  - Kan een hand detecteren op enkele centimeters van de sensor
  - Heel goedkoop te realiseren
- Hoe werkt het?
  - De sensorplaat (of oppervlak) en het menselijk lichaam vormen samen een capaciteit.
  - De grootte van de capaciteit is vooral bepalend van de afstand tussen je hand en de sensorplaat.
  - Door je hand dichterbij of verderweg te brengen van de sensor vergroot of verklein je de capaciteitswaarde waardoor de lading die het bevat ook wordt gewijzigd.
- Wat is de rol van Arduino?
  - In principe meet je met Arduino hoeveel tijd de capaciteit van de aanraaksensor nodig heeft om opgeladen te worden. Hierdoor kan een schatting gemaakt worden van de capaciteit. Deze kan zeer klein zijn maar wordt toch nog redelijk nauwkeurig bepaald.



3-3 Capacitieve sensor

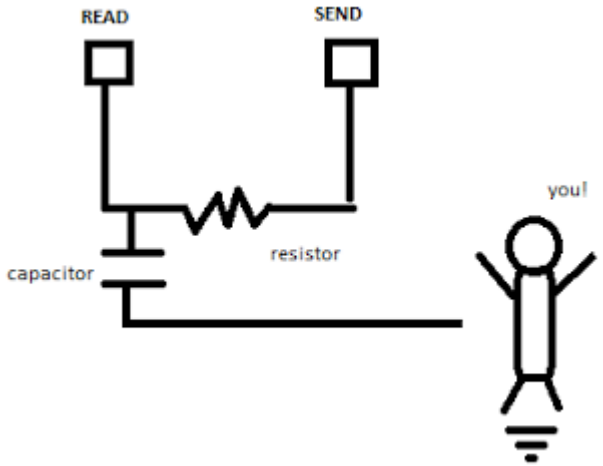
Tussen Send pin en Receive pin wordt een bepaalde weerstand geplaatst afhankelijk van de gewenste response.

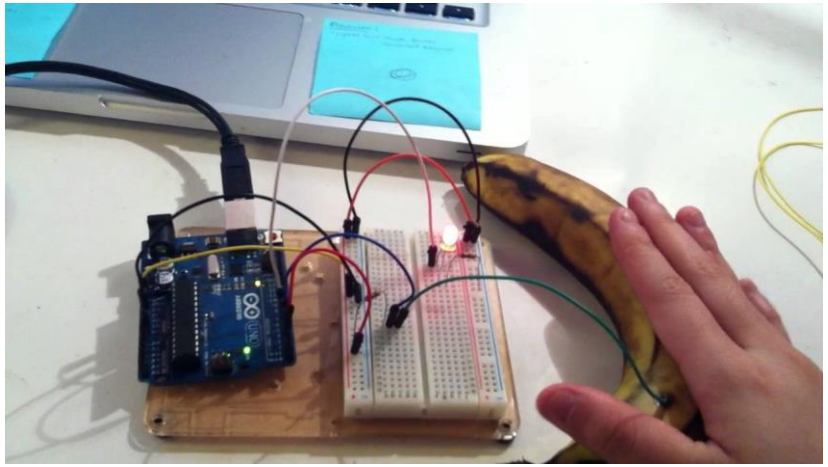
- 1 MΩ voor activatie bij aanraking
- 10 M Ω voor response op zo'n 10 à 15 cm afstand
- 40 M Ω voor response op zo'n 30 à 50 cm afstand

Hoe groter de weerstand, hoe groter de gevoeligheid van de sensor maar hoe trager de reactie zal zijn.

Een capaciteit van 1 nF in serie met de sensor kan eventueel 50 Hz bron voorkomen of sterk verminderen.

De weerstandswaarden gelden voor gebruik via de Capacitive Sensing bibliotheek.
http://playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense





3-3 Capacitieve sensor

Toepassing Capacitive Sensor

CapacitiveSensor Cs = CapacitiveSensor(send_pin ,receive_pin);

- Als send pin = 4 en receive pin =8 => CapacitiveSensor Cs = CapacitiveSensor(4,8);
- Cs is gewoon een gekozen naam voor de sensor

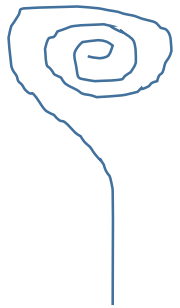
Serial.begin(9600);

- Stelt de datarate in (eenheid Baud) voor seriële communicatie

long v = Cs.capacitiveSensor(30);

- Bij gebruik van 1MΩ zal de variabele v minder dan of gelijk zijn aan 10 en bij aanraking wordt de waarde meer dan 60.

Opgave5 :
Maak naaststaande sketch en voer deze uit.
Gebruik al sensor een stuk draad met isolatie rond.
Rol het uiteinde op zoals in figuur is weergegeven.
Plaats een 10 M Ω-weerstand tussen pennen 4 en 8. Stel pin 8 in als sensorpin.
Probeer de afstanden te bepalen voor het weergeven van de verschillende tonen afzonderlijk.
Verklaar de code van de sketch en zet de sketch om in een flowchart ter verduidelijking van de werking



```
#include <CapacitiveSensor.h>

#include <CapacitiveSensor.h>


// capacitieve sensor
#include <CapacitiveSensor.h>

CapacitiveSensor Cs = CapacitiveSensor(4,8); // 10 Mohm tussen pennen
// pin 8 instellen als sensorpin
const byte luidspreker = 13; //audio uitgangskanaal
const int toonlengte = 100, del = 20; // toonlengte en vertraging
// toonfrequenties
const int A=440, B = 494;
const int C = 523, d = 587, e = 659, f=698;
const int g = 784, a = 880, b = 988, c = 1047;

const byte drempel = 30; // sensor-drempelwaarde

void setup() {
  pinMode(13, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  long v = Cs.capacitiveSensor(30);
  Serial.println(v); // stuur sensorwaarde naar PC
  if (drempel < v) {
    digitalWrite(13, HIGH);
    if ((v >= 40) & (v <= 50)) {tone(luidspreker, B, toonlengte); }
    if ((v >= 50) & (v <= 60)) {tone(luidspreker, c, toonlengte); }
    if ((v >= 60) & (v <= 70)) {tone(luidspreker, d, toonlengte); }
    if ((v >= 80) & (v <= 90)) {tone(luidspreker, e, toonlengte); }
    if ((v >= 90) & (v <= 100)) {tone(luidspreker, f, toonlengte); }
    if ((v >= 100) & (v <= 110)) {tone(luidspreker, g, toonlengte); }
    if ((v >= 110) & (v <= 120)) {tone(luidspreker, b, toonlengte); }
    if ((v >= 130))              {tone(luidspreker, c, 500); }
  }
  else digitalWrite(13, LOW);
  delay(del);
}
```