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
 };
```

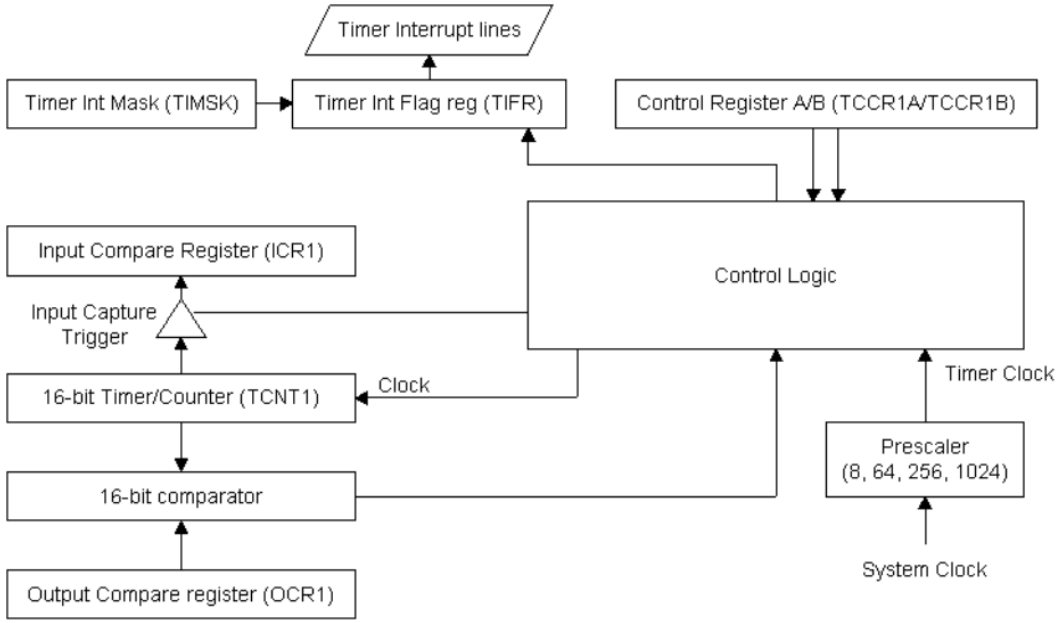
```
void setup() {
  pinMode(audioPin, OUTPUT);
  TCCR1A = 0b10000001;
  TCCR1B = 0b00001001;
 }

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

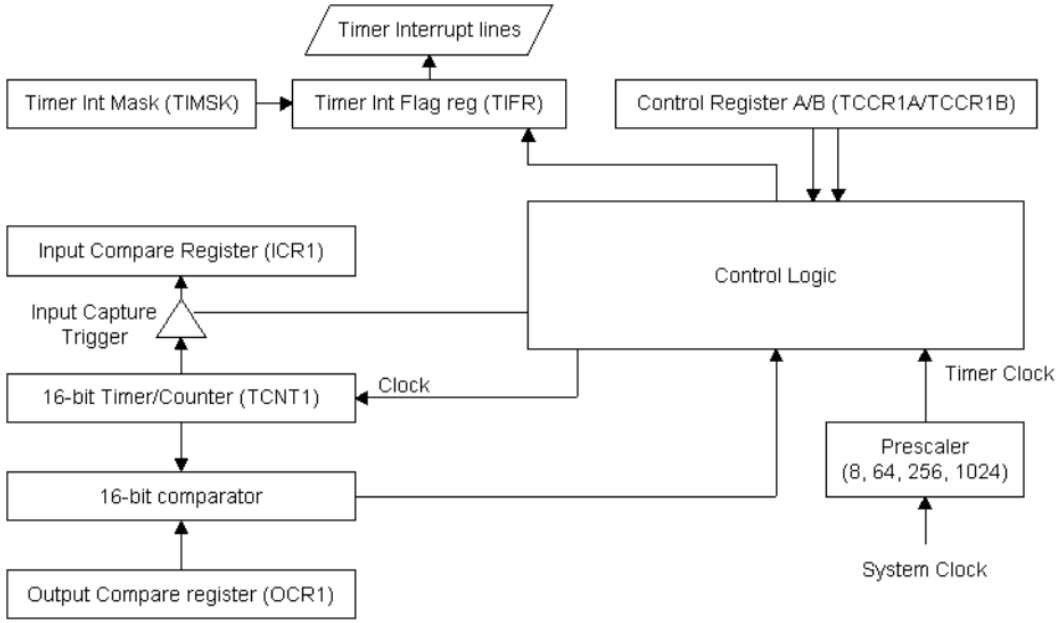
- Timers tellen normaal klokcycli. De klokcycli kunnen afkomstig zijn van de systeemklok of vertraagd worden door gebruik te maken van de prescaler (delen door 8, 64, 256 of 1024)
- Een 16-bit Timer heeft een bereik van 0xFFFF = 65536 telwaarden.
  - Stel dat de prescaler ingesteld staat op 1024 => 65536 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

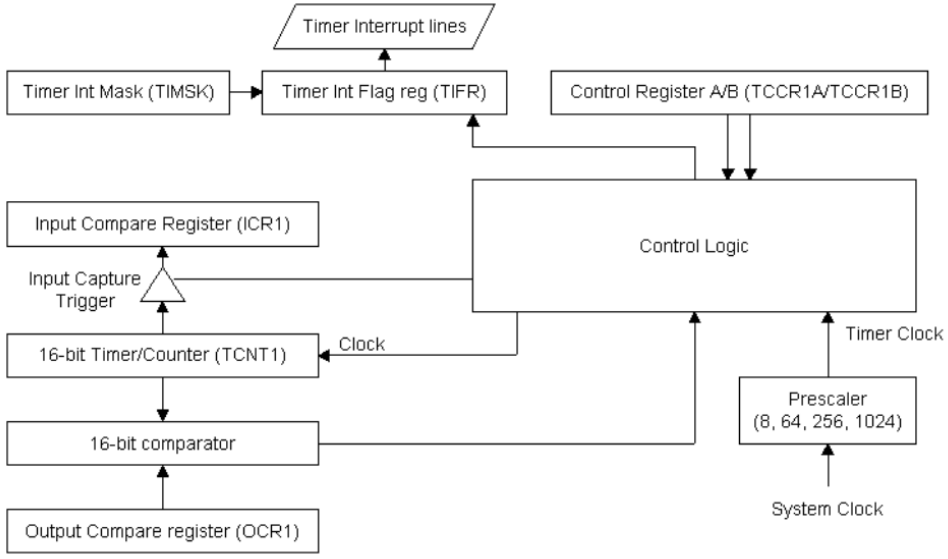
CS12	CS11	CS10	Mode Description
0	0	0	Stop Timer/Counter 1
0	0	1	No Prescaler (Timer Clock = System Clock)
0	1	0	divide clock by 8
0	1	1	divide clock by 64
1	0	0	divide clock by 256
1	0	1	divide clock by 1024
1	1	0	increment timer 1 on T1 Pin falling edge
1	1	1	increment timer 1 on T1 Pin rising edge

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);
}
```

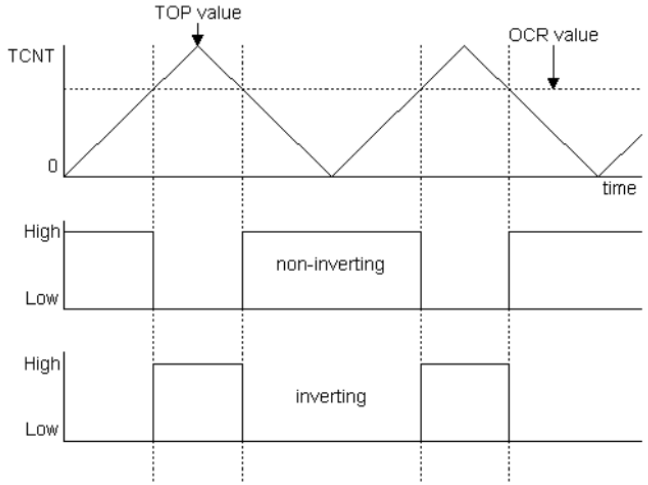
3-2 Geluiden met PWM

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.