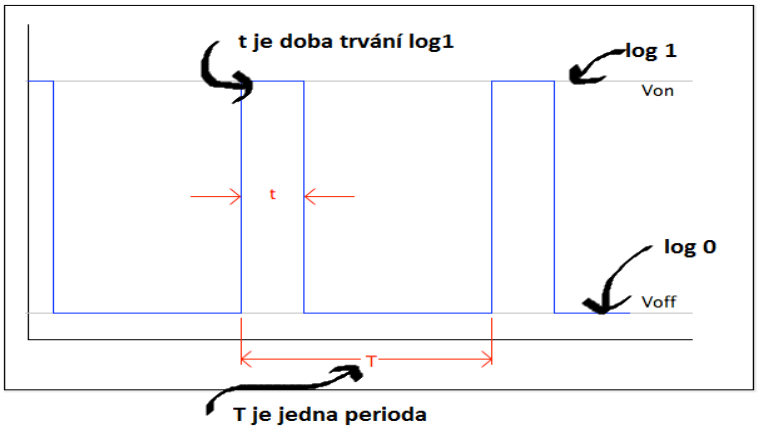
TEORIE PRO PWM LABORATORNÍ CVIČENÍ

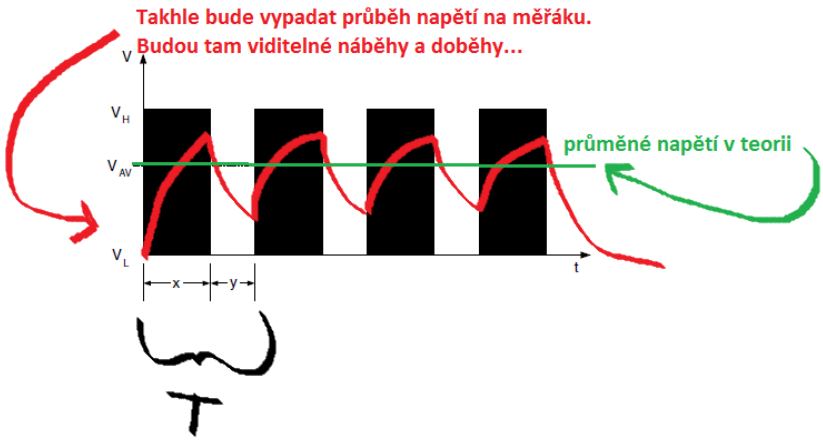
Začneme od toho jak v teorii funguje PWM (pulsně šířková modulace). Vyjde najevo, že je možné jí použít pro generování sinusového signálu. Také se zamyslíme nad tím co je nosná a generovaná frekvence a jak spolu tyto dvě frekvence souvisí.

ÚVOD DO GENEROVÁNÍ SINUSOVÉHO SIGNÁLU POMOCÍ PWM

V závislosti na délce trvání logické hodnoty 1 a logické hodnoty 0 během jedné časové periody/úseku se bude měnit průměrná hodnota napětí během této časové periody na výstupním pinu kde budeme sinusový signál odebírat.



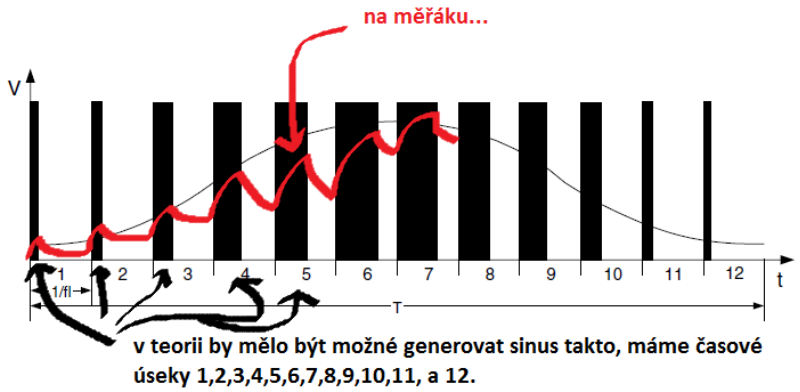
Jak by v teorii vypadalo generování průměrného napětí pomocí PWM je možné vidět také v aplikační poznámce 314 od AVR. Uvedu zde jen jejich lehce editovaný obrázek. Je to zhruba a od ruky takže tam chybí pár drobností, ale zatím je zamlčím.



Zde je značena perioda T. Jen tak mimochodem, perioda je něco co se opakuje, je možné to také chápat jako časový úsek, nebo časové kvantum. Takže zde je časové kvantum T rovné součtu doby trvání log 1 a doby trvání log 0, respektivě x+y. Takže průměrné napětí je možné obdržet podle následujícího vzrorce:

$$V_{AV} = \frac{x \cdot V_H + y \cdot V_L}{x + y}$$

V_{AV} je tedy průměrné napětí během úseku T. Je celkem snadné si domyslet, že pokud budeme vhodným způsobem měnit doby trvání log 1 v přilehlých časových úsecích, vytvoříme tím průběh napětí připomínající sinusový signál. Náš úmysl naznačuje následující obrázek.



Zbývá jen několik zatím nezodpovězených otázek, které je potřeba vyřešit než usedneme k programování. Uvedu je v následujícím seznamu:

1) Jak získáme dobu trvání logické 1 v časových úsecích jedné periody?
2) Jak změříme dobu trvání logické 1 na výstupním pinu?
3) Jaký je vztah mezi frekvencí generovaného sinusového signálu a frekvencí generujícího signálu?
4) Existuje omezení na maximální frekvenci signálu v otázce (3)?
5) Jak se zbavíme těch červených zubů?

Asi by mohlo snadno někoho napadnout i několik dalších otázek, ale zatím toho necháme, a zodpovíme ty co již máme. Zde jsou částečné odpovědi v pořadí, takže 1) je odpověd na otázku 1).

1) Dobu trvání logické jedničky v navazujících časových usecích získáme tím, že si předem vypočítáme vzorky jedné periody sinusu a uložíme je do tabulky ve flash paměti MCU.
2) Dobu trvání logické jedničky v jednom časovém useku změříme pomocí čítače zkonfigurovaného pro generování pulsně šířkové modulace na AVR procesoru.
3) Již z obrázku je partné, že generující PWM frekvence bude potřeba vyšší než frekvence generovaného sinusu.
4) Ano, ale vzhledem k tomu, že používáme čítač, který potřebuje dopočítat do velikosti každého vzorku, budeme značně dělit základní frekvenci krystalu na desce.
5) Navrhneme filtr prvního řádu pomocí odporu a kondenzátoru.

PODROBNOSTI KE KAŽDÉ ODPOVĚDI

Řešení a názor na zpracování otázky (1) se skládá z několika problémů. Konkrétně vypočítání vzorků sinusu a jejich uložení do flash paměti procesoru. Můžeme začít s úvahou nad řešením tím, že zavzpomínáme na funkci sinus, i.e., y(x) = sin(x), kde x je úhel, třeba α.

Jak je známo, jedna perioda funkce sinus se nachází na jednotkové kružnici mezi úhly 0 až 360°. Jediné co potřebujeme vědět k jejich vypočítání je požadovaný počet vzorků N_c a jejich rozlišení. Rozlišeni udáváme v bitech, takže rozlišení **n** znamená že mezi napětím V_L a V_H bude $\mathbf{2^n}$ hladin napětí. Vzhledem k tomu, že Atmega16 je 8-bitový stroj, budeme pracovat s maximálně 8-bitovým rozlišením.

Předběžná úvaha jak získat tyto vzorky by asi probíhala tak, že potřebujeme postupně zvyšovat hodnotu úhlu α od nuly po 360° po stejných skocích tak, aby počet těchto skoků byl roven počtu vzorků $\mathbf{N_c}$, a pro každý skok vypočítat funkční hodnotu **sin(α).** Označíme-li skok (pravidelný přírůstek) Δ_α pak vzniknou následující závislosti:

$$\Delta_\alpha = \frac{360}{N_c}$$

A vzorek **S** s indexem **i**, tedy $\mathbf{S_i}$ a rozlišením **n** bude mít velikost:

$$S_i = \left\lceil 2^n \cdot \sin\left(i \cdot \Delta_\alpha\right) \right\rceil$$

Zbývá nám pouze dořešit jak to udělat, aby $0 \subseteq \mathbf{S_i} \subseteq \mathbf{2^n}$, to znamená, aby vzorky byly větší nebo rovné nule a menší nebo rovné $\mathbf{2^n}$, kde n je rozlišení u našeho 8-bitového procesoru, tudíž n je maximálně 8. S tím si ale snadno poradíte.

Další problém je jak uložíme vzorky do tabulky:

```
;************************** SIN TABLE ***********************************
; Samples table : one period sampled on 128 samples and
; quantized on 7 bits
;************************************************************************
sine_tbl:
.db 64,67
.db 70,73
.db 76,79
.db 82,85
.db 88,91
.db 94,96
.db 99,102
```

Jinak tohle je jen příklad, vaše vzorky asi budou vypadat uplně jinak.

Řešení a názor na zpracování otázky (3) a (4). Nebo-li jak souvisí PWM frekvence s frekvencí generované sinusovky. Je zřejmé, že budeme generovat sinusový signál analogový o určité

modulované frekvenci, **f**. Dále je zřejmé, že máme na základní desce krystal frekvence $\mathbf{f_{ck}}$, kterou nelze překročit, protože by došlo k přetaktování procesoru.
Je také pravda, že budeme používat určitou frekvenci generujícího pulně šířkového signálu $\mathbf{f_{PWM}}$. Samozdřejmě nemůžeme zapomenout na počet vzorků $\mathbf{N_c}$ a jejich rozlišení **n**. Rozlišení budeme požadovat pokud možno co nejvyšší, snad 8-bitů, ale jsme omezeni rychlostí procesoru.

Zatím to vypadá, že co se týče proměnných, máme **f**, $\mathbf{f_{ck}}$, $\mathbf{f_{PWM}}$, **n**, a počet vzorků. Jako zadání obdržíme pravděpodobně jen **f**. Vhodně můžeme také použít dělení $\mathbf{f_{ck}}$ podle datasheet, v našem případě, no prescale, /8, /64, /512, /1024.

Náš první krok bude vycházet z aplikační poznámky AVR314, a to rovnice (4). Tam se dozvíme, že modulovaná frekvence je rovná modulující PWM frekvenci vydělené počtem vzorků.

$$f = \frac{f_{PWM}}{N_c} \Rightarrow \frac{f_{PWM}}{f} = N_c$$

Takže počet vzorků pak odvodíme z této rovnice snadno.

Další dilema, které si asi zaslouží zamyšlení je souvislost mezi počtem vzorků a jejich rozlišení.

Je celkem přirozené očekávat, že při vyšším rozlišení budu chtít mít větší počet vzorků, ale pak budu nucen uložit do flash více dat, a budu schopen generovat nižší maximální modulovanou frekvenci **f**, protože jsem omezen rychlostí procesoru.

Naopak při nižším rozlišení mi stačí nižší počet vzorků, ale budu schopen generovat potenciálně vyšší modulovanou frekvenci **f**. Nechte si tyto dvě myšlenky projít hlavou a pochopíte, že modulující frekvenci $\mathbf{f_{PWM}}$ pak můžeme odhadnout podle následujícího vzorce, kde **N** je dělení frekvence základní frekvence krystalu, $\mathbf{2^n}$ je počet vzorků s rozlišením **n** a $\mathbf{f_{ck}}$ je základní frekvence krystalu na desce.

$$f_{PWM} = \frac{f_{ck}}{2 \cdot N \cdot 2^n}$$

Nyní vyzkoušíme naše nápady na nějakém zadání, například tomto: Vygenerujte pomocí procesoru sinusový analogový signál s frekvencí **f** = 770Hz. Máte procesor Atmega16 osazený krystalem s frekvencí $\mathbf{f_{ck}}$ = 12Mhz. Věrohodnost signálu by měla být co nejlepší.

Na základě našich předchozích úvah je možné udělat předběžné zvážení o tom jak by souvisela generující $\mathbf{f_{PWM}}$ frekvence s generovanou frekvencí **f**.

Dejme tomu, že na základní desce máme krystal 12Mhz, požadujeme 8-bit rozlišení vzorků a čistě náhodně se rozhodneme, že budeme dělit základní frekvenci krystalu osmi (prescale factor). Pak máme $\mathbf{f_{PWM}} = 12x10^6/(2x8x256) = 2929Hz$.

Nebo dejme tomu, chceme 7-bit rozlišení, no prescale, 12MHz Xtall, generujte 770Hz sinus. Pak máme $\mathbf{f_{PWM}} = 12x10^6/(2x127) = 46785Hz$

Pokud budeme v úvaze pokračovat budeme chtít generovat analogový sinus s frekvencí 770Hz, kolik vzorků bude potřeba? Vezmu data z příkladu jedna.

Snadno tento počet odhadneme tím že vydělíme $\mathbf{f_{PWM}}$ /**f** = 2929/770 = 4, což vychází žalostně málo. Netvrdím, že je nemožné vygenerovat analogový sinus se čtyřmi vzorky, ale zrekonstruovaný s těmito parametry asi bude velmi nevěrohodný.

Je zřejmé že čtyři vzorky přesto, že jsou v rozlišení 8-bit, nevygenerují věrohodnou sinusovku. Co s tím?

Můžeme udělat dvě věci. (1) nedělit základní frekvenci krystalu a (2) snížit rozlišení vzorků.

Zkusíme nejdříve možnost (1) což je konfigurace bez prescale, tím dostaneme 8x4=32 vzorků. Což je ještě trochu málo.

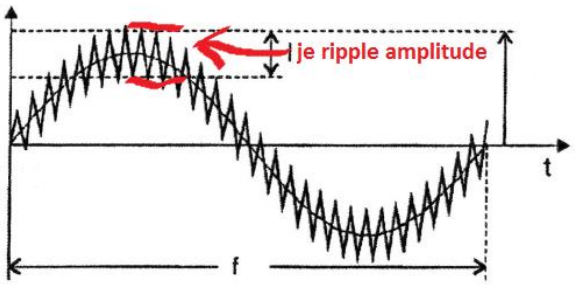
Zkusíme rozlišení 7-bit. Tím dostaneme zhruba 60 vzorků. Kdybychom se pokusili o rozlišení 6-bit dostali bychom zhruba 120 vzorků, ale asi by to byl kompromis, ale roche jiného rázu. Nechám vás vysvětlit proč.

Dejme tomu, že se rozhodneme pro 60 vzorků sinusu. Jak je vygenerujeme? 360/60 = 6, takže náš úhel budeme inkrementovat od 0 do 360 po 6 stupních. Doporučuji si udělat jednoduchý program například v JavaScript na generování vzorků $\mathbf{S_i}$ a to rovnou s **.db**, aby to šlo snadno překopírovat do AVR studia.

JAK NAVRHNOUT FILTR

Pro tuto část budeme potřebovat trochu terminologie a teorii. Nejdříve terminologie.

1) Filtr **dolní propust** redukuje vysoké frekvence a nechá projít nízké frekvence
2) **Zesílení** je míra zvětšení amplitudy signálu, $V_{out}/V_{in} > 1$.
3) **Atenuace A** nebo-li **útlum** $V_{out}/V_{in} < 1$.
4) **Zlomová frekvence $\mathbf{F_c}$** je u filtru dolní propust ta frekvence ve které se **atenuace** začíná zvyšovat se zvyšující se frekvencí.
5) **Decibel** je systém jak měřit **útlum** a **zesílení**. Pro natětí je definován jako **Db**=20log(Vout/Vin).
6) **Ripple** zvlnění je amplituda generující pwm frekvence, která projde filtrem.



Ripple frekvenci chceme odfiltrovat, nebo **značně** snížit její amplitudu. Zatím je všeobecně známo, že největší amplituda tohoto zvlnění je při účiníku (duty cycle) 50%.

Nejdůležitější **<u>otázka číslo jedna</u>** je jaký útlum filtru je potřeba pro určení maximální apmlitudy (ripple) zvlnění. K tomu použijeme následující vzoreček, se kterým to odhadneme:

$$V_{rip} \cong S \cdot A \cdot \frac{\pi}{2}$$

To je pro filtr prvního řádu. Pro filtr druhého a třetího řádu použijeme následující odhad:

$$V_{rip} \cong S \cdot A \cdot \frac{4}{\pi}$$

1) maximální ripple napětí, které chceme označíme $\mathbf{V_{rip}}$
2) peak-to-peak PWM napětí označíme **S**. (Na našem procesoru je to 0 až 5V)
3) atenuace (útlum) označíme **A**.

V podstatě tyto dva vzorce je možné pochopit tak, že se jakoby ptáme, jaký útlum filtru je potřeba, když chceme maximální ripple napětí $\mathbf{V_{rip}}$ a maximální rozsah generujícího PWM napětí **S**?

Vzhledem k tomu, že se pokoušíme navrhnout filtr prvního řádu z kondenzátoru a odporu, možná jsou naše požadavky nereálné. Rozumný požadavek na filtr prvního řádu by se dal považovat útlum nejvíce –20dB na dekádu. Filtr druhého a třetího řádu dosáhnou lepších parametrů.

Například se můžeme zeptat následově: maximální zvlnění, které jsem schopen tolerovat pro můj filtr prvního řádu je $\mathbf{V_{rip}} = 0.1V$, rozsah generujícího PWM signálu **S = 5.**0V, vypočtěte útlum, který je potřeba k vyhovění parametrů.

K řešení použijeme první vzorec, takže po lehké machinaci dostaneme $\mathbf{A} \cong (0.1/5)x(2/\pi) = 0.0127$, respektivě v dB = 20log(0.0127) = -38dB. Což je možná trochu víc útlumu než je filtr prvního řádu splnit. Asi bychom se spokojili s větším zvlněním než 0.1V.

Nejdůležitější **<u>otázka číslo dva</u>** je kolik Hz je lomová frekvence $\mathbf{F_c}$? Na odhad odpovědi použijeme následující vzoreček:

$$F_c = A \cdot F_{PWM}$$

Pokud použijeme předchozí parametry pro 7-bit rozlišení, bez dělení základní frekvence krystalu, pro generování sinusu s frekvencí 770Hz, Po dosazení obdržíme $\mathbf{F_c = A \cdot f_{PWM}} = 0.0127 \cdot 46875 = 595Hz$.

Co to znamená? To znamená, že pokud chceme generovat sinus s frekvencí **f** < 595Hz filtr prvního řádu bude asi možný. Takže pro frekvenci 770Hz projde filtrem prvního řádu tolik zvlnění, že se rozpadne integrita modulovaného sinusu.

Abychom to napravili, použili bychom filtr druhého řádu. Ten by měl odhadnutou atenuaci ~40dB na dekádu a asi by vyhověl (měli jsme požadavek –38dB).

Frekvence lomu pro filtr druhého řádu by se pak odhadla podle následujícího vzorečku:

$$F_c = F_{PWM} \cdot \sqrt[2]{A}$$

takže $\mathbf{F_c} = 46875 \cdot \sqrt{0.0127} = 5282Hz$, což je více než 770Hz, takže filtr druhého řádu by nám vyhověl. Já do konstrukce filtru druhého a vyššího řádu ale nepůjdu.

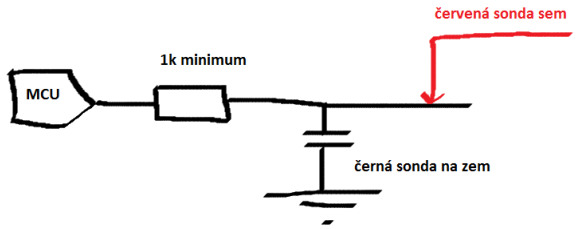
KONSTRUKCE RC FILTRU PRVÍHO ŘÁDU

Jak již název napovídá jedná se o filtr z odporu a kondenzátoru. Jen tak mimochodem, není dobrý nápad použít kondenzátor s jakýmkoliv dialektrikem, použijte foliový. Další věc je použít minimální hodnoty 1k pro odpor a 220pF pro kondenzátor. Co dál?

Použijeme následující vzoreček:

$$RC = \frac{1}{2\pi F_c}$$

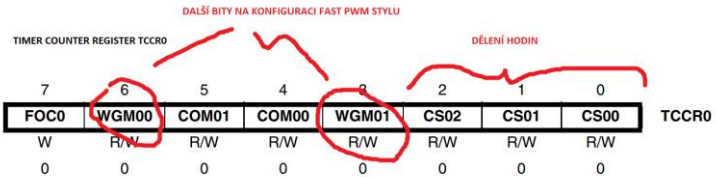
Další otázka je jak zapojíme odpor a kondenzátor a jak to připojíme k procesoru? Schéma bude vypadat zhruba následovně:



JAK VYPADÁ KONFIGURACE PROCESORU?

Použijeme fast PWM konfiguraci periferie, kterou má náš chip již integrovanou. Co to znamená je, že jako obvykle vyhledáme potřebné konfigurační údaje v datasheetu. Tyto údaje se skládají z (1) konfiguračních registrů a (2) konfiguračních hodnot.

Náš konfigurační registr je TCCR0 s následujícím rozložením konfiguračních bitů:



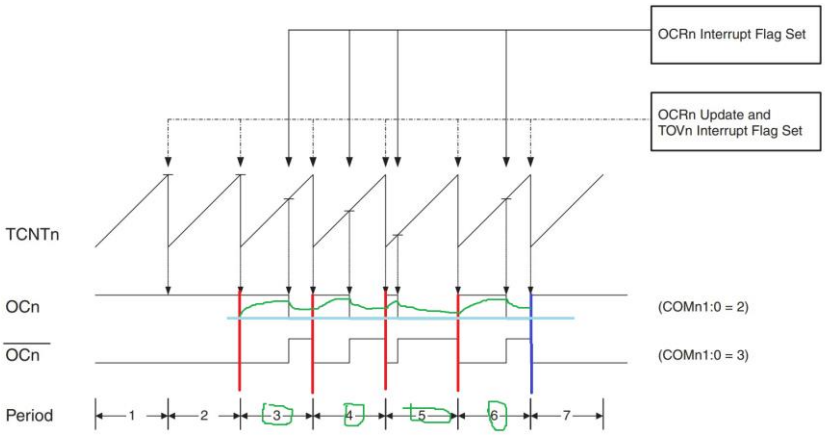
Bity CS snad už nemusím vysvětlovat. Další nové konfigurační bity, se kterými se seznámíme jsou WGM bity. Tyto bity pomáhají ztvarovat průběh signálu podle následující tabulky:

Table 38. Waveform Generation Mode Bit Description(1)

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

Note: 1. The CTC0 and PWM0 bit definition names are now obsolete. Use the WGM01:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Po tomto způsobu konfigurace se čítač bude chovat následovně:



Je patrné, že napětí se bude měnit podél zelené, takže v levé části půjde spíš nahoru, pak pozvolna dolů ve střední části a potom opět nahoru v pravé části.

Další drobnost, které si povšimneme, je že výrobce nám již dodal konfigurační možnost periferie čítače, kde se výstup (OCn) v dané periodě přepne z log. 1 na log. 0 podle námi zkonfigurované vrchní hodnoty a doběhne do plné délky periody na logické nule. Tím vznikne požadovaný duty cycle (účiník) pro danou periodu.

VYPRACOVANÝ PŘÍKLAD

Použijeme naši obvyklou školní desku s ATmega16, osazenou Xtalem s nominální frekvencí 14.7456Mhz. Použijeme 7-bit rozlišení, nedělíme základní frekvenci Xtalu.

Podle předchozích odhadů je $\mathbf{f_{PWM}} = 14.7456 \times 10^6/(2 \times 127) = 58.053$kHz. Nyní zbývá otázka kolik vzorků budeme potřebovat. Počet vzorků se bude odvíjet podle frekvence, kterou

budeme generovat. Například při 128 vzorcích vygenerujeme 58053/128=453Hz. Je zřejmé, že touto cestou (dopočítaváním vzorků) máme vyšší flexibilitu.

Jakou zvolíme atenuaci? Volíme $\mathbf{V_{rip}}$ = 0.1V, pak podle předchozích výpočtů (pokud zachováme rozsah napětí generující PWM frekvence 0 až 5V) dostaneme -38dB. Můžeme se dostat pod tento nereálný požadavek tím, že dovolíme vyšší ripple zvlnění (třeba 0.2V). Nechám vás domyslet kolik by to mělo být.

Z toho vyplývá zlomová frekvence $\mathbf{F_c = A \cdot f_{PWM}} = 0.0127 \cdot 58053 = 737.3$Hz. To vypadá dobře, protože tato zlomová frekvence je vyšší než 453Hz, kterou chceme generovat, takže generovaná filtrem prvního řádu projde a 58kHz již ne.

Navrhneme filtr, $RC=1/(2 \cdot \pi \cdot 737.3) = 2.1586 \cdot 10^{-4}$. S tímto údajem se zamyslíme nad hodnotou kondenzátoru a rezistoru. Začneme dosazovat hodnoty odporu.

1k znamená $C = 2.1586 \cdot 10^{-4} \cdot 10^{-3} = 215$nF
2k znamená $C = 2.1586 \cdot 10^{-4} \cdot 10^{-3} / 2 = 107$nF
4k znamená $C = 2.1586 \cdot 10^{-4} \cdot 10^{-3} / 4 = 53$nF
4.7k znamená $C = 2.1586 \cdot 10^{-4} \cdot 10^{-3} / 2 = 46$nF

Vyberu odpor 4.7k, protože jej mám a kondenzátor 46nF, protože mám kondenzátor 22nF a hodnotu 44nF mohu snadno složit ze dvou.