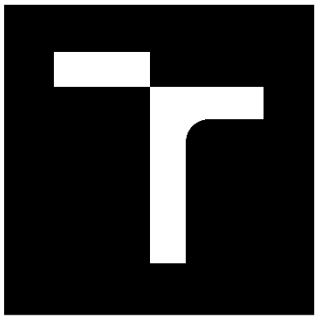


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

MIKROKONTROLÉREM ŘÍZENÉ ZAPALOVÁNÍ
IGNITION SYSTEM CONTROL BY MICROCONTROLER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
Dominik Drahoš

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Tomáš Macho, Ph.D.

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Automatizační a měřicí technika**
Ústav automatizace a měřicí techniky

Student: Dominik Drahoš ***ID:*** 165825
Ročník: 3 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Mikrokontrolérem řízené zapalování

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s problematikou řízení zapalování spalovacích motorů.
2. Navrhněte koncepci systému elektronického zapalování řízeného mikrokontrolérem. Nakreslete blokové schéma systému.
3. Zvolte vhodný mikrokontrolér. Navrhněte obvodové řešení systému elektronického zapalování, nakreslete schéma zapojení a stanovte hodnoty jednotlivých součástek.
4. Navrhněte desku plošných spojů, vytvořte výrobní dokumentaci.
5. Zařízení realizujte.
6. Vytvořte potřebné softwarové vybavení.
7. Ověřte funkčnost zařízení a zhodnoťte dosažené výsledky.

DOPORUČENÁ LITERATURA:

[1] FROHN, Manfred - OBERTHÜR, Wolfgang - SIEDLER, Hans-Jobst - WIEMER Manfred - ZASTROW, Peter. Elektronika - polovodičové součástky a základní zapojení. Praha: BEN 2006. 500 s. ISBN 80-7300-123-3.

[2] Vlastimul V., Zapalování a seřizování předstihu. Dostupné z: http://www.velorexy.cz/clanek/zapalovani-a-serizovani-predstihu

Termín zadání: 8.2.2016 ***Termín odevzdání:*** 23.5.2016

Vedoucí práce: Ing. Tomáš Macho, Ph.D.
Konzultant bakalářské práce:

doc. Ing. Václav Jirsík, CSc., *předseda oborové rady*

ABSTRAKT

Tato práce se zabývá návrhem mikrokontrolérem řízeného zapalování. V úvodu jsme seznámeni s principem zapalování spalovacích motorů. Dále obsahuje seznam existujících typů zapalování. Na konec vyobrazuje kompletní postup návrhu obvodového řešení elektronického zapalování a jeho softwarové řešení.

KLÍČOVÁ SLOVA

Elektronické zapalování, tranzistorové zapalování, řízení předstihu, návrh zapalování

ABSTRACT

This thesis describes the design of ignition system controlled by microcontroller. In the introduction, we are familiar with the principle of ignition combustion engines. It also contains a list of existing types ignition. Finally portrays the entire design process circuitry electronic ignition and software solution.

KEY WORDS

Electronic ignition, transistor ignition, timing control, ignition proposal

Bibliografická citace:

DRAHOŠ, D. *Mikrokontrolérem řízené zapalování.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. XY s. Vedoucí bakalářské práce Ing. Tomáš Macho, Ph.D..

Prohlášení

„Prohlašuji, že svou diplomovou (*bakalářskou*) práci na téma *Mikrokontrolérem řízené zapalování* jsem vypracoval samostatně pod vedením vedoucího diplomové (*bakalářské*) práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové (*bakalářské*) práce dále prohlašuji, že v souvislosti s vytvořením této diplomové (*bakalářské*) práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: ……

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Tomáši Machovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: ……

podpis autora

Obsah

1. ÚVOD

Každý zážehový motor potřebuje ve vhodný okamžik zapálit palivovou směs. Dle teorie, která se vyučuje na základních školách, by směs měla být zapálena v horní úvrati, tj. ve zlomovém bodu pohybu pístu (nejblíže k zapalovací svíčce). To ovšem platí pouze pro nízké otáčky, jelikož doba hoření směsi není závislá na otáčkách. Je závislá např. na bohatosti, teplotě. Můžeme tedy říct, že doba hoření je téměř konstantní, zatím co doba trvání otáčky se při jejich zvyšování snižuje. Je tedy nutné zapalovat směs již před horní úvratí. Úhel, o který je zapálení posunuto před horní úvrať se nazývá předstih. Pevným nastavením předstihu motor funguje optimálně jen v určitých otáčkách, případně v malém rozsahu. Pokud tedy chceme provozovat motor ve velkém rozsahu otáček, je vhodné předstih regulovat dle otáček.

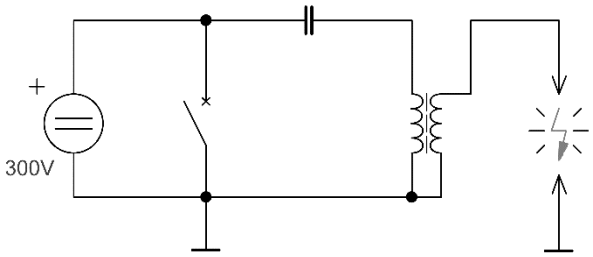
Regulace otáček probíhá zpožďováním řídícího signálu ze snímače natočení. Snímačem může být indukční snímač, optická závora, nebo v mém případě Hallova sonda, která je umístěna tak abychom získali hranu signálu v určité vzdálenosti před horní úvratí. Z hlediska programování ideálně v 90°, jelikož je to čtvrtina otáčky. Budeme-li tedy měřit dobu otáčky časovačem, můžeme při nastavení čtvrtinové předděličky u druhého časovače pouze zkopírovat hodnotu z prvního časovače. Posun menší než 90° lze lehce kompenzovat programem. Tento způsob však funguje pouze při malé změně periody z otáčky na otáčku, proto je hallovu sondu vhodné doplnit o inkrementální snímač, který je tvořen optickou závorou a diskem s vhodným počtem impulzů na otáčku (v mém případě 360 -> přesnost 1°)

2. DRUHY ZAPALOVÁNÍ

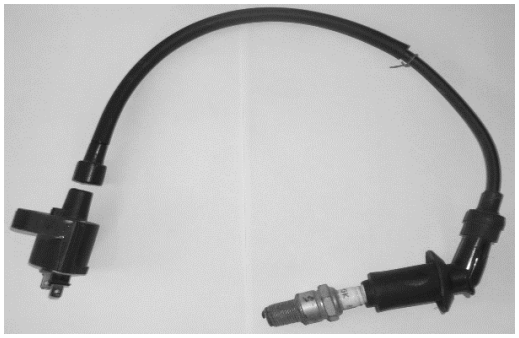
Kapacitní zapalování (CDI)

Zapalovací cívka je pouze transformátor, na který se přivádí energie z kondenzátoru. Tento kondenzátor se nabíjí napětím vyšším než má baterie (200-300V). Toto napětí může být získáno přímo z alternátoru motoru (AC-CDI), nebo generováno měničem z baterie (DC-CDI)

DC-CDI je sice náročnější na návrh a výrobu, ale zaručí nezávislost síly jiskry na otáčkách. Díky tomu se zlepší funkce motoru při nižších otáčkách.



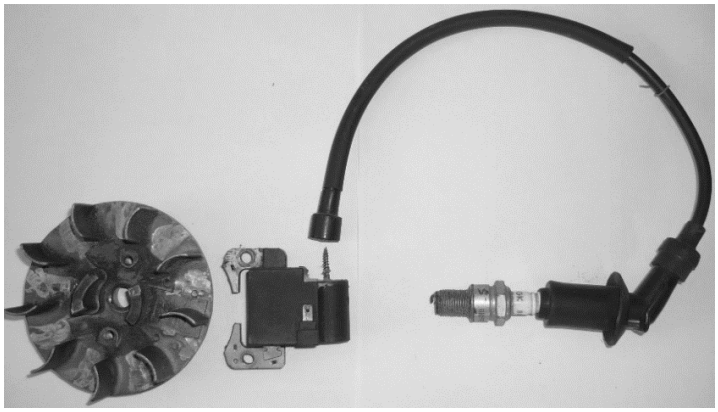
Obr. 2.1 Schématické zapojení CDI



Obr. 2.2 Zapalovací cívka pro CDI

Magnetoelektrické zapalování

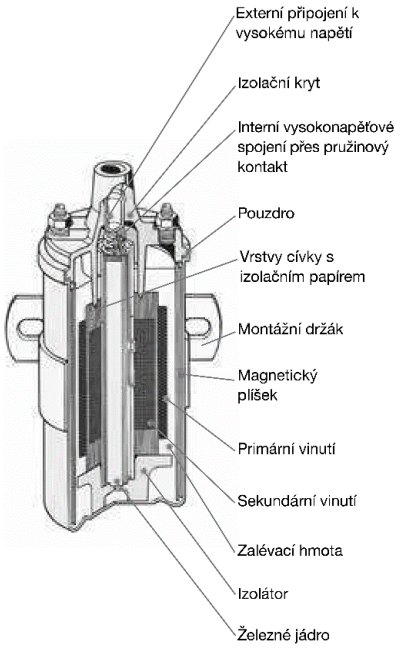
Setrvačník s permanentními magnety je přímo připojen ke klikovému hřídeli, se kterým se otáčí. Na bloku motoru je připevněna cívka s jádrem z transformátorových plechů. Pohyb magnetů v blízkosti jádra cívky na ní indukuje napětí, které je dále zesíleno transformátorem, který může být umístěn v pouzdře společně s cívkou, nebo zvlášť. Z transformátoru vystupuje již vysoké napětí pro připojení ke svíčce.



Obr. 2.3 Magnetoelektrické zapalování

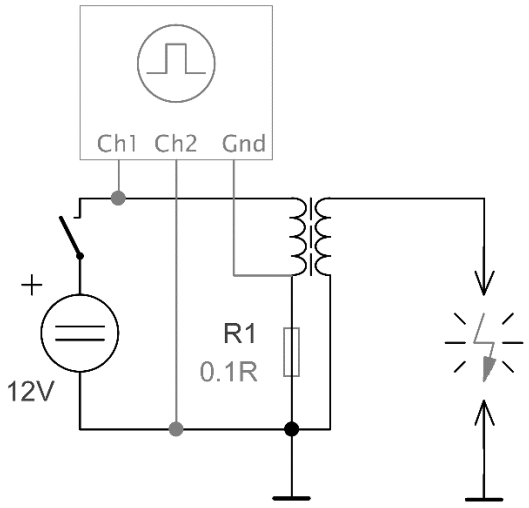
Bateriové zapalování

Toto zapalování je velmi často používáno. Energie je ukládána přímo do těla cívky. Nepotřebuje tedy kondenzátor ani zvýšené napětí. Při připojení cívky k nominálnímu napětí (většinou 12V, ale může být i 6V) začne proud protékající cívkou narůstat k ustálené hodnotě (chová se jako klasický RL článek). Po odpojení napájení se uvolní energie uložená v cívce do sekundárního obvodu, ve kterém se transformuje na vysoké napětí. Toto napětí se při přivedení na svíčku změní na jiskru.



Obr. 2.4 Průřez zapalovací cívkou (převzato z [1])

Obr. 2.5 Zapalovací cívka bateriového zapalování

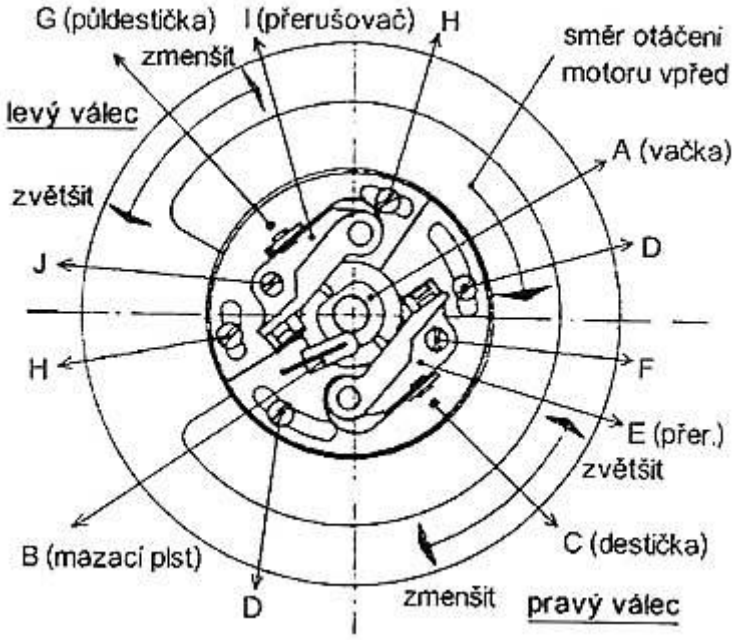


Obr. 2.6 Schématické zapojení bateriového zapalování

Bateriové zapalování lze dělit dle způsobu spínání:

Kladívkové:

Na klikovém hřídeli je připevněna vačka, která mechanicky spíná a rozepíná vodivý kontakt. Na jednu stranu kontaktu je připojeno napájení z baterie a na druhý je připojen na cívku. Při odpojení proudu do cívky ale vzniká elektrický oblouk a dochází k opalování kontaktů. Proto je paralelně ke kontaktu připojen kondenzátor, který pohlcuje energii oblouku.



Obr. 2.7 Kladívkové spínání cívek (převzato z [2])

Elektronické:

Napětí je připojováno k cívce pomocí tranzistoru, nebo jiného spínacího prvku. Odpadá tedy problém s opalováním kontaktů. Zároveň je možné snadno měnit předstih zpožďováním řídícího signálu tranzistoru. Proto jsem si zvolil tento typ zapalování k realizaci své práce.

3. NÁVRH ZAPALOVÁNÍ

Výběr mikrokontroléru (uC)

Při výběru mikrokontroléru jsem se řídil následujícími atributy:

a) Výrobce

Při výběru jsem si mohl vybírat z mnoha výrobců, jako jsou Atmel, Freescale, Microchip, SMT…

Rozhodl jsem se pro Microchip, protože jejich mikrokontroléry používám, tudíž vím, jak fungují a co od nich mohu očekávat.

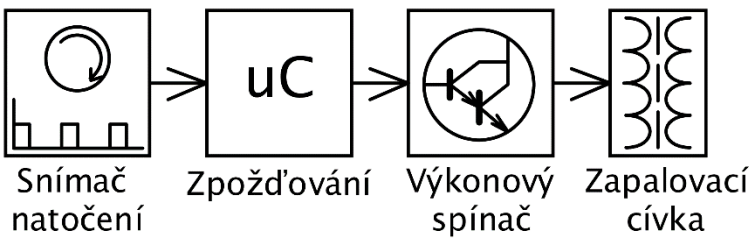
b) Rychlost

Dalším atributem výběru byla rychlost mikrokontroléru. Ta je velmi důležitá, protože výpočet zpoždění z křivky předstihu může být časově náročné. Díky tomu jsem se rozhodl pro řadu PIC18, která sice není úplně nejrychlejší, ale s rychlostí 12 MIPS (miliónů instrukcí za sekundu) a díky hardwarové násobičce bohatě postačí i pro 20000 otáček za minutu.

c) Komunikace s PC.

Posledním parametrem mého výběru bylo komunikační rozhraní pro nahrání křivky předstihu. Zde jsem mohl volit mezi rozhraními USB a RS232. USB je dostupné v každém dnešním počítači, zatím co pro RS232 bych musel používat převodník a umístit napěťový převodník MAX232 na desku plošných spojů, čímž zvětšil velikost desky. Proto jsem se rozhodl pro USB 2.0. Nejlepší možností mi přišlo použít kontrolér, který bude přímo podporovat USB, čímž ušetřím prostor na desce.

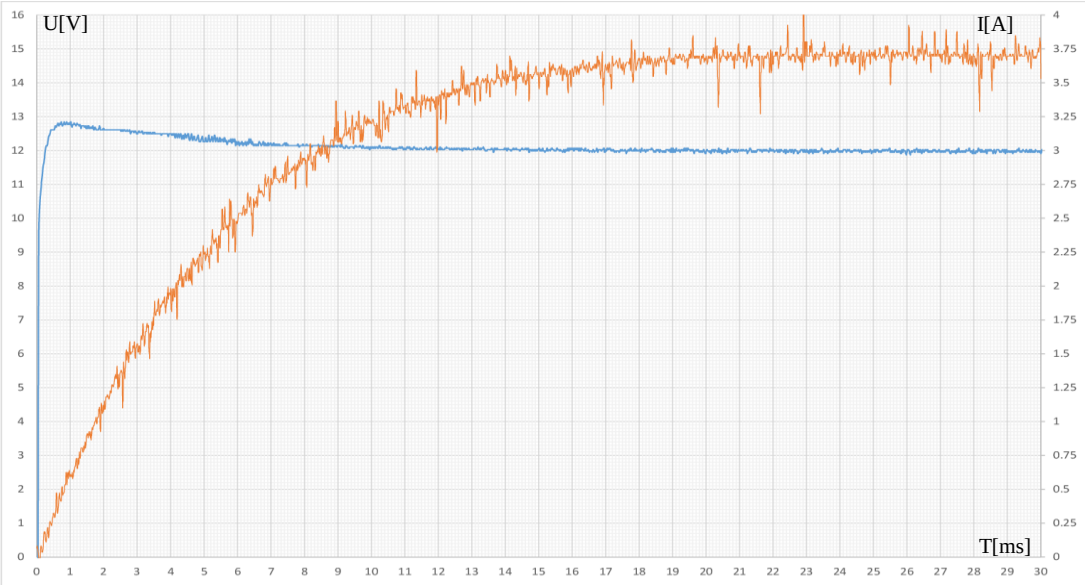
Blokové schéma zapalování



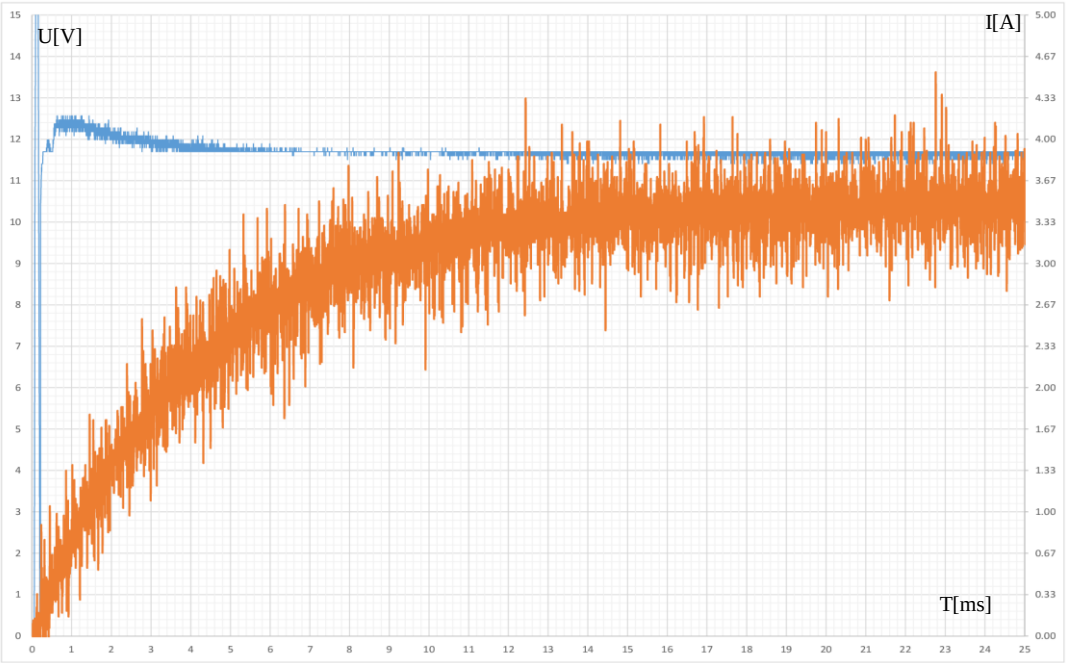
Obr. 3.1 Blokové schéma TCI

Identifikace zapalovacích cívek

Pro návrh výkonového výstupního členu zapalování je potřeba nejprve identifikovat zapalovací cívky. K určení parametrů cívek jsem je zapojil dle obr 2.6 a změřil přechodovou charakteristiku. Z ní jsem potom určil maximální proud cívkou a časovou konstantu. Z ustáleného proudu a napětí na cívce jsem spočítal odpor cívky. Na závěr pak z časové konstanty a odporu i indukčnost cívky.

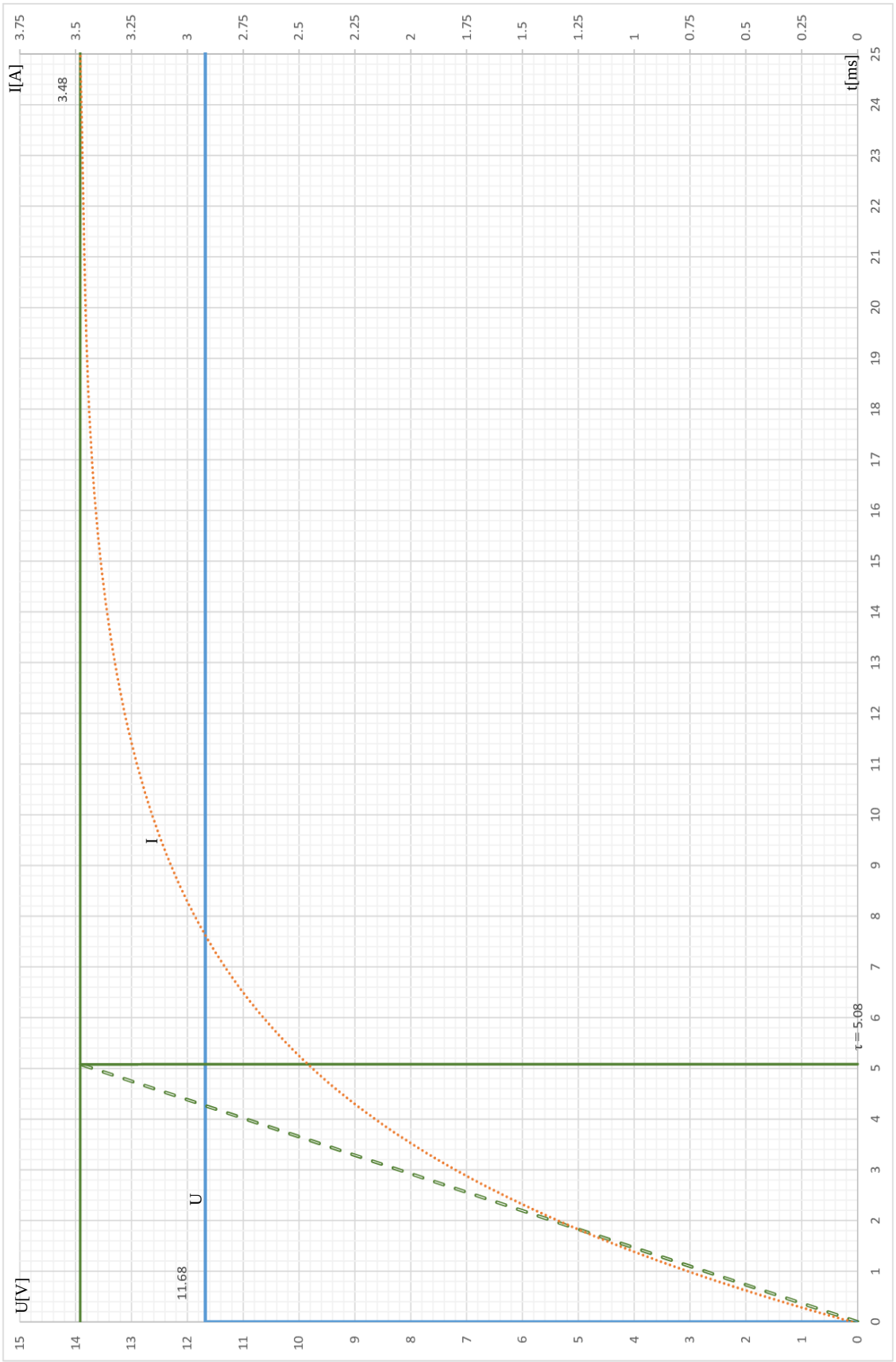


Obr. 3.2 Naměřená charakteristika zapalovací cívky Java



Obr. 3.3 Naměřená charakteristika zapalovací cívky Škoda

Obr. 3.4 Proložení charakteristiky cívky Java pro zjištění parametrů



Obr. 3.5 Proložení charakteristiky cívky Škoda pro zjištění parametrů

Parametry cívek (vyčteno z charakteristik):

Cívka Java:	Cívka Škoda:
Ustálené napětí U = 11.98V	Ustálené napětí U = 11.68V
Ustálený proud I = 3.72A	Ustálený proud I = 3.48A
Časová konstanta τ = 6.48ms	Časová konstanta τ = 5.08ms
Odpor vinutí $R = \frac{U}{I} = \frac{11.98}{3.72} = 3.22\Omega$	Odpor vinutí $R = \frac{U}{I} = \frac{11.68}{3.48} = 3.36\Omega$
Indukčnost $L = \tau * R = 3.22 * 6.48E-3 = 20.9mH$	Indukčnost $L = \tau * R = 3.36 * 5.08E-3 = 17.07mH$

Tab. 3.1 Změřené parametry zapalovacích cívek

Při běhu motoru je napětí na baterii zvýšeno alternátorem na 13.5V. Tím se zvýší i maximální proud cívky, ten bude větší u cívky Java, jelikož má menší odpor.

Maximální proud cívkou $I_{MAX} = \frac{U_{MAX}}{R_{Java}} = \frac{13.5}{3.22} = 4.19A$

Návrh výkonového spínacího stupně

Volba spínacího tranzistoru

Pro návrh tranzistoru je vhodné maximální proud naddimenzovat alespoň 1.5x => I_{max} = 4.19*1.5 = 6.29A

Dále je potřeba určit efektivní a střední hodnotu proudu. K tomu je nutné znát střídu proudu. Tu určíme z maximálních otáček, které jsem se rozhodl určit na N = 20 000 min^{-1} a doby nabíjení, kterou jsem experimentálně a na základě [2] určil na t_{nab} = 1.5ms.

Perioda proudu $T = \frac{60}{N} = \frac{60}{20000} = 3ms$

Střída $s = \frac{t_{nab}}{T} = \frac{1.5}{3} = 0.5$

$I_S = I_{max} * s = 6.29 * 0.5 = 3.15A$

$I_{ef} = I_{max} * \sqrt{s} = 6.29 * \sqrt{0.5} = 4.45A$

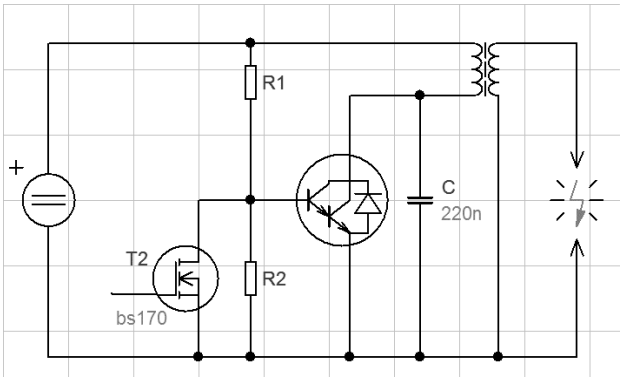
Potřebujeme tedy tranzistor na spínání 6.29A. Při vyhledávání vhodného tranzistoru jsem narazil na bu931 – darlingtonův tranzistor pro řízení zapalování „High voltage ignition coil driver NPN power Darlington transistor“ [3], který bude nejvhodnější, jelikož je přizpůsoben na vysokonapěťové překmity vznikající při odpojení zapalovací cívky od napájení.

Parametry tranzistoru BU931T

Pouzdro: TO-220	P_{max} = 125W
U_{CEmax} = 500V	ϑ_{jmax} = 175°C
I_{Ccont} = 10A	$R_{\vartheta jc}$ = 1.2°C/W
I_{Bcont} = 1A	h_{fe} = 300

Nyní, když známe parametry tranzistoru, můžeme navrhnout zapojení. Výstupy uC není vhodné přetěžovat, proto jsem se rozhodnul přidat ještě jeden předřadný unipolární tranzistor BS170. Výpočtem přes R_{DSon} jsem ověřil, že bude tranzistor dobře spínat. Jelikož se jedná o stejný výpočet jako zde uvedený pro ověření odporů, ale k odporu R2 se připočítá paralelní odpor R_{DSon} = 1.2Ω

Zapojení:



Obr. 3.6 Schéma zapojení výkonového členu

Výpočet součástek výkonového členu

Tranzistor T1 ovšem nemá v sepnutém stavu nulový odpor. Je tedy nutné přepočítat proud cívkou. Z hodnot U_{CEsat} pro různé kolektorové proudy jsem si spočítal proložení, čímž jsem se dopustil drobné chyby. Ta není tak velká, aby výpočty ovlivnila.

$$U_{CEsat} = 0.2 * I_{CM} + 0.2$$

$$U_Z = U - U_{CE} => R_Z * I_{CM} = U - 0.2 * I_{CM} - 0.2 => I_{CM} = \frac{U - 0.2}{R_Z + 0.2} = \frac{13.5 - 0.2}{3.22 + 0.2}$$

$$= 3.89A * 1.5 = 5.83A$$

Hodnotu jsem opět vynásobil koeficientem 1.5, abych získal rezervu. Z charakteristiky (Fig. 4 [3]) jsem pro tento proud vyčetl proudové zesílení β (h_{fe}) = 250.

Nyní můžeme vypočítat proud báze.

$$I_B = \frac{I_C}{\beta} = \frac{5.83}{250} = 23.32mA \triangleq 25mA$$

Dále je nutné zjistit saturační napětí báze emitor. To jsem opět zjistil z datasheetu proložením hodnot.

$$U_{BEsat} = 0.2 * I_{CM} + 0.8 = 0.2 * 5.83 + 0.8 = 1.97V$$

Nyní můžeme vypočítat hodnoty odporů R1 a R2.

$$U_{BEsat} = I_{R2} * R_2 => I_{R2} = \frac{U_{BEsat}}{R_2}$$

$$I_B = I_{R1} - I_{R2} = \frac{U - U_{Bsat}}{R_1} - \frac{U_{Bsat}}{R_2} =>$$

$$=> R_1 * (I_B * R_2 + U_{BEsat}) = R_2 * (U - U_{BEsat}) =>$$

$$=> R_1 = R_2 * \frac{(U - U_{BEsat})}{(I_B * R_2 + U_{BEsat})} = R_2 * \frac{13.5 - 1.97}{25E-3 * R_2 + 1.97} = \frac{11.53 * R_2}{25E-3 * R_2 + 1.97}$$

Tímto jsme získali jednu rovnici o dvou neznámých. Ta má nekonečně mnoho řešení. Nyní si musíme dát pozor na další parametr tranzistoru a to maximální napětí báze emitor, které je dle datasheetu 5V. Zvolím si tedy trošku menší hodnotu U_{BE} = 4.5V

$$U_{BE} = U * \frac{R_2}{R_1 + R_2} => R_2 = \frac{R_1 * U_{BE}}{U - U_{BE}} = \frac{R_1 * 4.5}{13.5 - 4.5} = 0.5 * R_1$$

Nyní již máme soustavu dvou rovnic o dvou neznámých. Dosazením jedné do druhé můžeme dospět k řešení.

$$R_2 = 0.5 * R_1 = \frac{0.5 * 11.53 * R_2}{25E-3 * R_2 + 1.97} = \frac{5.77 * R_2}{25E-3 * R_2 + 1.97}$$

$$25E - 3 * R_2^2 + 1.97 * R_2 - 5.77 * R_2 = 0$$

$$R_2 * (0.025 * R_2 - 3.8) = 0$$

$$R_2 = 152\Omega$$

Porovnáním odporu s normovanými hodnotami zvolím nejbližší hodnotu tj.: 150Ω.

Z druhé rovnice:

$$R_2 = 0.5 * R_1 => R_1 = 2 * R_2 = 2 * 150 = 300\Omega$$

Ověření hodnot

Nyní pro jistotu ověříme vypočítané odpory.

$$U_{BE} = U * \frac{R_2}{R_1 + R_2} = 13.5 * \frac{300}{150 + 300} = 4.5V$$

$$I_B = I_{R1} - I_{R2} = \frac{U - U_{Bsat}}{R_1} - \frac{U_{Bsat}}{R_2} = \frac{13.5 - 1.97}{300} - \frac{1.97}{150} = 25.3mA$$

Zde nám vyšly žádané hodnoty. Ještě by bylo dobré ověřit funkčnost při vybité baterii. Počítejme nejnižší přípustnou hodnotu olověného 6-ti článkového akumulátoru, která je 10.5V

$$I_{CM} = \frac{U - 0.2}{R_Z + 0.2} = \frac{10.5 - 0.2}{3.22 + 0.2} = 3.01A * 1.5 = 4.51A$$

Pro tento proud je hodnota zesílení přibližně β = 350. Na bázi tedy musíme přivést proud

$$I_B = \frac{I_C}{\beta} = \frac{4.51}{350} = 12.9mA.$$

Saturační napětí báze emitor pro tento proud Ic vychází:

$$U_{BEsat} = 0.2 * I_{CM} + 0.8 = 0.2 * 4.51 + 0.8 = 1.702V$$

Při ponechání hodnot odporů vychází proud I_B

$$I_B = I_{R1} - I_{R2} = \frac{U - U_{Bsat}}{R_1} - \frac{U_{Bsat}}{R_2} = \frac{10.5 - 1.702}{300} - \frac{1.702}{150} = 17.98mA$$

Hodnota proudu vypočítaná z odporů je větší, než potřebná hodnota k úplnému otevření tranzistoru, takže zapalování bude fungovat i na minimální hodnotu baterie.

Návrh chladiče

Nejprve přepočítáme efektivní a střední hodnoty proudu pro cívku s tranzistorem.

$$I_S = I_{CM} * s = 5.83 * 0.5 = 2.92A$$

$$I_{ef} = I_{CM} * \sqrt{s} = 5.83 * \sqrt{0.5} = 4.12A$$

Nyní potřebuji spočítat ztrátový výkon na tranzistoru T1. Pro určení ztrátového výkonu na bipolárním tranzistoru se používá poučka pro činný výkon na konstantním odporu.

$$u = R_{CEon} * i$$

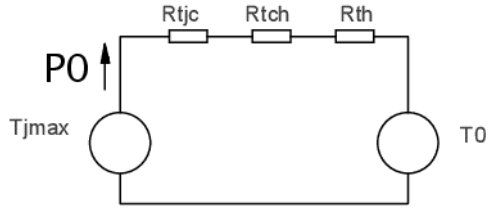
$$U_{CEsat} = 0.2 * I_{CM} + 0.2 = 0.2 * 5.83 + 0.2 = 1.366V$$

$$R_{CEon} = \frac{U_{CEsat}}{I_{CM}} = \frac{1.366}{5.83} = 0.23\Omega$$

$$P = R_{CEon} * {I_{ef}}^2 = 0.23 * 4.12^2 = 3.9W$$

K určení chladiče potřebujeme už jen teplotní odpor mezi pouzdrem a chladičem. Ten je dle dat na [5] $R_{\vartheta ch}$ = 3°C/W.

Nyní si můžeme nakreslit náhradní schéma tepelného obvodu:

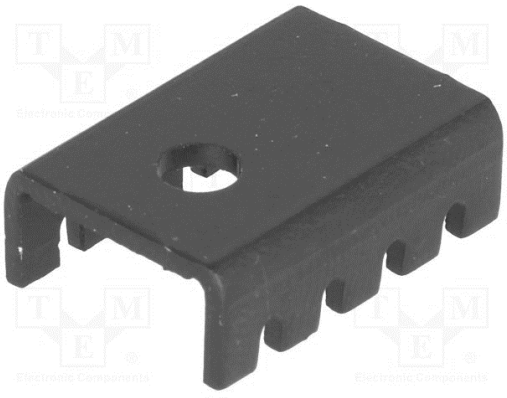


Obr. 3.7 Náhradní schéma tepelného obvodu

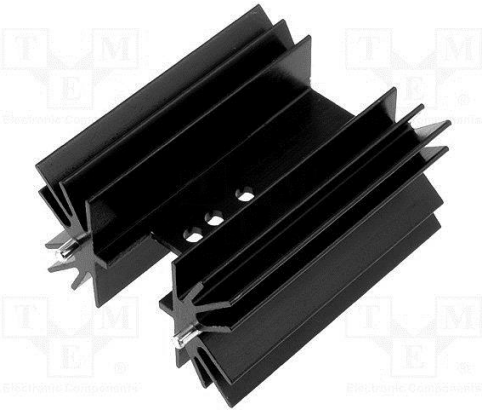
$$-T_{Jmax} + P_0 * \left(\mathrm{R}_{\vartheta \mathrm{jc}} + \mathrm{R}_{\vartheta \mathrm{ch}} + \mathrm{R}_{\vartheta \mathrm{h}}\right) + T_0 = 0$$

$$\mathrm{R}_{\vartheta \mathrm{h}} = \frac{T_{Jmax} - T_0}{P_0} - \mathrm{R}_{\vartheta \mathrm{jc}} - \mathrm{R}_{\vartheta \mathrm{ch}} = \frac{175 - 40}{3.9} - 3 - 1.2 = 30.4°C/V$$

K uchlazení tranzistoru by stačil chladič minimálních rozměrů z lisovaného plechu. Např.: HS-S01 [6] o rozměrech 19.05*13.21*6.35mm. Při takto malém chladiči by ale teplota přechodu byla téměř přesně 175°C což je maximální teplota přechodu. Proto je vhodné zvolit větší kvalitnější chladič např.: HS-142-50 [7] s teplotním odporem 5.2°C/W, při kterém by teplota přechodu byla pouze 76°C a teplota chladiče 60.2°C.



Obr. 3.8 Chladič HS-S01 (Převzato z [6])



Obr. 3.9 Chladič HS-142-50 (Převzato z [7])

Obvod mikrokontroléru

Pro získání maximální rychlosti zvolíme nejrychlejší krystal, který je možné připojit k uC. To je 20MHz [4]. Pro tento krystal opět vyčteme z datasheetu velikost kondenzátorů (Tab. 2-2 [4]) = 15pF. Dále musíme připojit reset (MCLR) na +5V. To je kvůli programování nutné udělat přes rezistor, který se pro CMOS logiku volí 10kΩ.

Nyní připojíme USB konektor a zvolíme kondenzátor pro napěťový regulátor pro USB (pin V_{USB}), který je dle datasheetu 220nF (TABLE 28-5 [4]). Zapojení ještě doplníme děličem připojeným na analogový vstup ke hlídání stavu baterie.

Pro dělič se vstupním napětím 13.5V a výstupním 5V nám vyjde poměr $R_1=1.7*R_2$. Pokud si zvolíme proud děličem 1mA vyjdou nám hodnoty odporů $R_1=8500\Omega$ a $R_2=5000\Omega$. Hodnota R_1 není v žádné odporové řadě. Zvolíme tedy nejbližší tj.: 8.66kΩ.

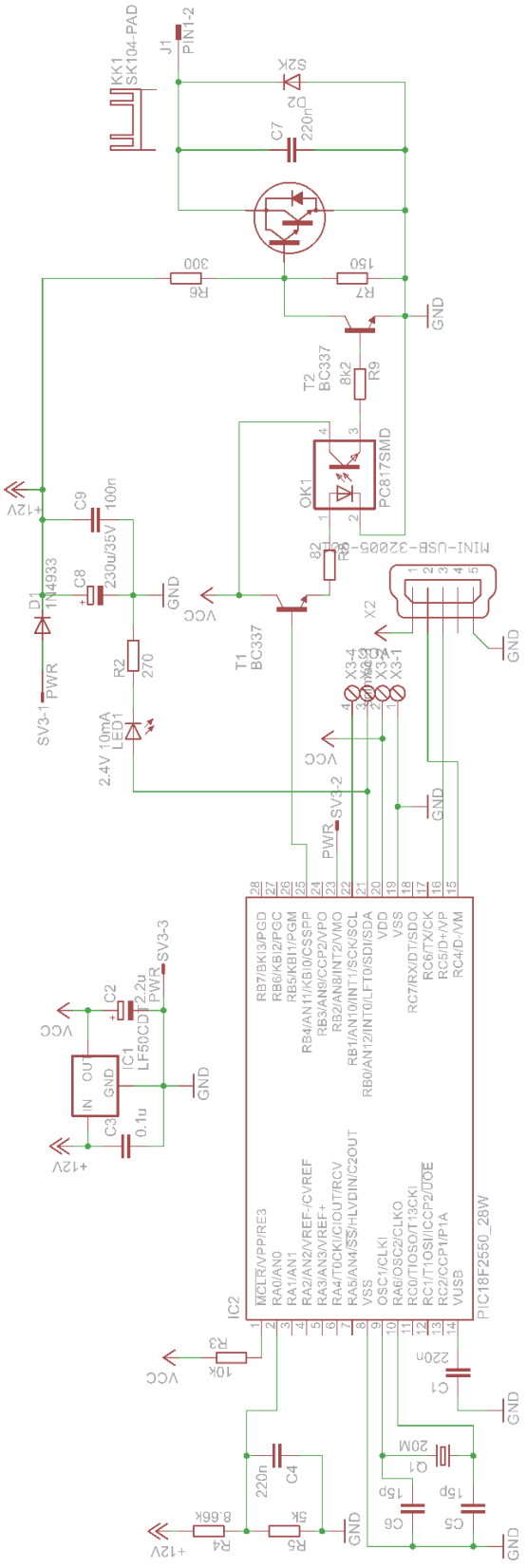
Poslední část zapojení je stabilizátor napětí pro uC. Zde použijeme LF50CDT. 5-ti voltový stabilizátor s maximálním proudem 0.5A v pouzdře DPACK. Hodnoty Kondenzátorů vyčteme opět z datasheetu[9]. Mají hodnoty $C_{in} = 0.1\mu F$ a $C_{out} = 2.2\mu F$.

Zapojení jsem ještě doplnil o diodu D1 proti přepólování napájecího napětí, dva filtrační kondenzátory na vyfiltrování napájecího napětí (velikost jsem určil ze zkušenosti), LED pro zobrazení řídícího signálu výkonového obvodu, optočlen pro galvanické oddělení mikrokontroléru od výkonového obvodu a odpor R1 pro Hallovu sondu ke snímání otáček motoru [10].

Odpor R8 pro diodu optronu byl určen z pracovního napětí a proudu jeho diody.

Teď již můžeme nakreslit kompletní schéma zapojení.

Schéma zapojení

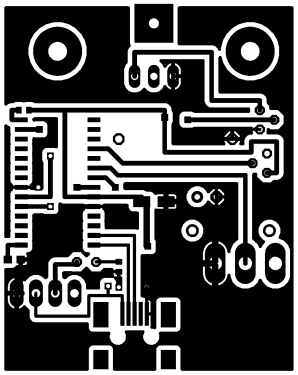


Obr. 3.10 Kompletní schéma zapojení zapalování

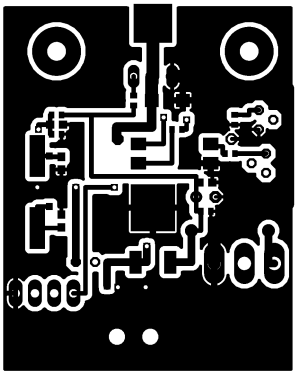
Deska plošného spoje

Z kompletního schématu je již možné nakreslit desku plošného spoje. Rozhodl jsem se pokud možno využít součástky pro povrchovou montáž a navrhnout výslednou DPS co nejmenší (je levnější na výrobu). Na desku jsem zahrnul i chladič HS-142-50 pro výkonový tranzistor, čímž jsem dosáhl větší odolnosti proti jeho případnému ulomení.

Měděné cesty:

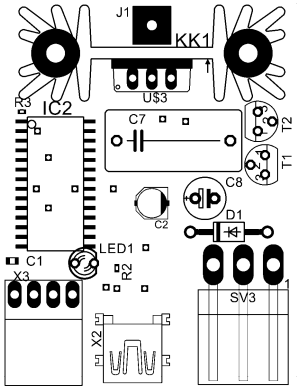


Obr. 3.11 Horní strana DPS

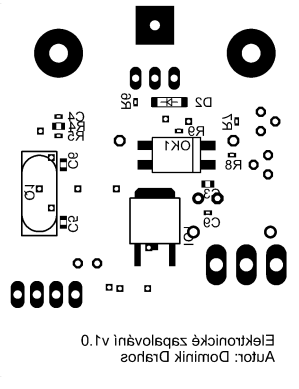


Obr. 3.12 Spodní strana DPS

Rozmístění součástek:



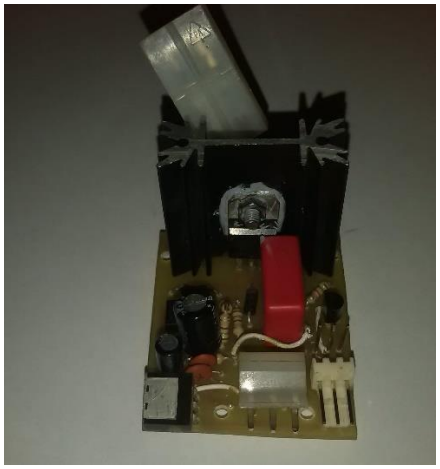
Obr. 3.13 Osazovací výkres horní strany



Obr. 3.14 Osazovací výkres spodní strany

4. REALIZACE ZAPALOVÁNÍ

Při realizaci zapalování jsem nejprve vytvořil desku plošných spojů, která obsahovala pouze část obvodu s optočlenem a stabilizátor napětí (Obr 4.1). Na vstupní konektor byl přiveden přímo výstup z hallovy sondy. Takto připravené zapojení jsem připojil k motoru (Obr 4.2) a otestoval jsem jeho funkci. Tím jsem zjistil, že výkonová část funguje dle předpokladu. Dále jsem připojil mikrokontrolér, ve kterém byl program kopírující vstup na výstup, umístěný na jiné desce a po několika drobných komplikacích s restartováním uC, způsobených indukováním napětí na rezistoru pro MCLR (stačilo jej vyměnit za SMD), jsem motor úspěšně rozeběhl.



Obr. 4.1 Výkonová část obvodu

Obr. 4.2 Výkonová část připojena k motoru

Dalším krokem bylo navrhnout programové vybavení pro HW a pro PC

Výpočet předstihu:

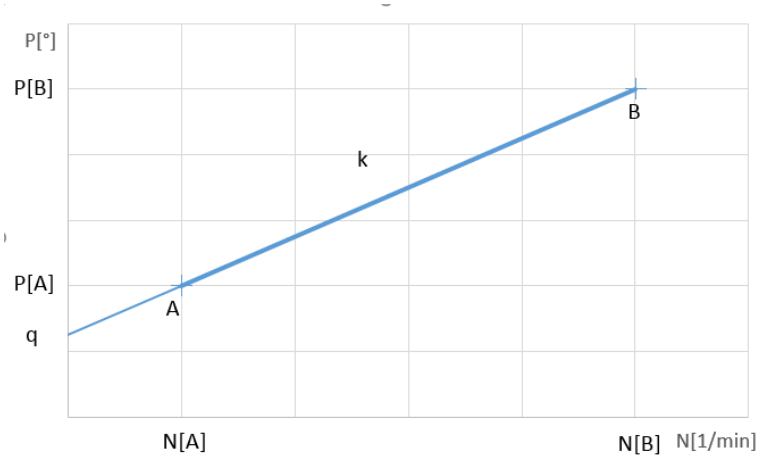
Předstih jsem se rozhodl počítat z doby otáčky. Hallův snímač jsem umístil do 90° před horní úvrať (tj. 270°). Výstup ze snímače vyhodnocuji přerušením reagujícím na náběžnou hranu. V přerušení vždy odečtu hodnotu časovače a vynuluji jej. Tuto hodnotu odečtu od maximální hodnoty čítače (2^{16}-1) a výsledek vložím do 4x rychlejšího čítače. Tím se dostanu do horní úvrati. Od této hodnoty ještě odečtu hodnotu vypočítanou z křivky předstihu, čímž se dostanu na požadovaný předstih.
Pro zrychlení výpočtu probíhají všechny přepočty v počítači a mikrokontrolér již pracuje pouze s hodnotami čítačů. Křivka předstihu je při tom uložena ve formě úseček popsaných pomocí směrnice a posuvu. Výsledná křivka je omezena na 25 dílčích úseček. Toto omezení je dáno velikostí vnitřní paměti EEPROM, do které je křivka uložena. Pro měření aktuální hodnoty otáček je použit čítač TMR0 s 16x předděličkou a pro nastavení výstupního signálu čítač TMR1 s 4x předděličkou. S takto nastavenou předděličkou je zde omezení minimálních otáček na 687 1/min, což ale nevadí, jelikož volnoběžné otáčky jsou přibližně 3000 1/min

Přepočet otáček na hodnotu čítače:

$$TMR0 = \frac{f_{ins} * 60}{N * Pr} = \frac{12 * 10^6 * 60}{N * 16} = \frac{45 * 10^6}{N}$$

f_{ins} – Vnitřní frekvence uC (v mém případě 12MHz)
N – Otáčky
Pr – Předdělička použitého čítače (16)

Nyní je potřeba vypočítat zpoždění, které budeme odměřovat pomocí čítače TMR1. To se spočítá z křivky předstihu rozdělené na lineární úseky. Výpočet tedy probíhá vždy ze dvou bodů A[N, P] a B[N, P], které se přepočítají do čítačové roviny:
A[TMR, P[TMR]] a B[TMR, P[TMR]]
Hodnota zpoždění je zde počítána jako část hodnoty čítače TMR0



Obr. 4.3 Jeden úsek křivky předstihu pro výpočet

$$P[TMR] = \frac{P}{90} * TMR$$

$$k = \frac{P(B)-P(A)}{N(B)-N(A)} = \frac{P[TMR](B)-P[TMR](A)}{TMR0(B)-TMR0(A)} = \frac{\frac{P(B)}{90}*TMR0(B)-\frac{P(A)}{90}*TMR0(A)}{TMR0(B)-TMR0(A)} =$$

$$= \frac{P(B)*TMR0(B)-P(A)*TMR0(A)}{90*(TMR0(B)-TMR0(A))}$$

$$P = k * x + q => q = P - k * x = \frac{P(A)}{90} * TMR0(A) - k * TMR0(A) =>$$

$$q = \left(\frac{P(A)}{90} - k\right) * TMR0(A)$$

Nyní máme spočítanou křivku předstihu a uloženou v paměti uC. Změříme dobu otáčky pomocí TMR0 a pomocí křivky vypočítáme hodnotu pro zpoždění:

$$TMR1 = 2^{16} - TMR0 + k * TMR0 + q + 18$$

Číslo 18 je konstanta kompenzující dobu výpočtu změřená při debugování.

uC:

Program pro mikrokontrolér jsem se rozhodl napsat ve vývojovém prostředí mikroc, ve kterém je díky knihovnám programování velice jednoduché.

Program po spuštění načte z paměti EEPROM počet dílčích křivek a postupně načte jednotlivé křivky do paměti RAM, aby se zamezilo čekání na načtení hodnot během výpočtů. Ke čtení hodnot z paměti je použita knihovní funkce

unsigned char EEPROM_Read(unsigned char addr)

Každá křivka se skládá z 10 Bytů:

2B - značí do jaké hodnoty čítače TMR0 je daná část křivky platná. Podle této hodnoty se taky vybírá aktuální křivka.

2*4B – Směrnice a ofset dané křivky (k a q) uložené ve formátu float

Rozložení paměti EEPROM:

Count	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk
LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm
LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq
Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk
Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq
LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk
LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm
LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq
Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk
Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq
LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk
LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm
LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq
Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk
Hk	Lk	LLk	HHq	Hq	Lq	LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq
LLq	HTm	LTm	HHk	Hk	Lk	LLk	HHq	Hq	Lq	LLq					

Obr. 4.4 Rozložení paměti EEPROM

Count – Celkový počet křivek které budou použity. Slouží pro zrychlení přenosu. Pokud budeme mít křivku o dvou dílčích úsecích, je zbytečné přepisovat zbytek EEPROM nulami.

HT a **LT** – horní a spodní Byte hodnoty čítače, do které platí aktuální křivka.

HH, **H**, **L**, **LL** – Byty jednotlivých float čísel pro k a q seřazené od nejvyššího po nejnižší.

Číslo typu float je u procesorů Microchip dle normy Microchip AN575. Ta je sice kompatibilní s IEEE754 (používanou v PC), ale má mírnou odlišnost v umístění znamínkového bytu.

IEEE-754 (PC)

	8b	23b
|S|	Exp	Mantisa
31	30 23	22 0

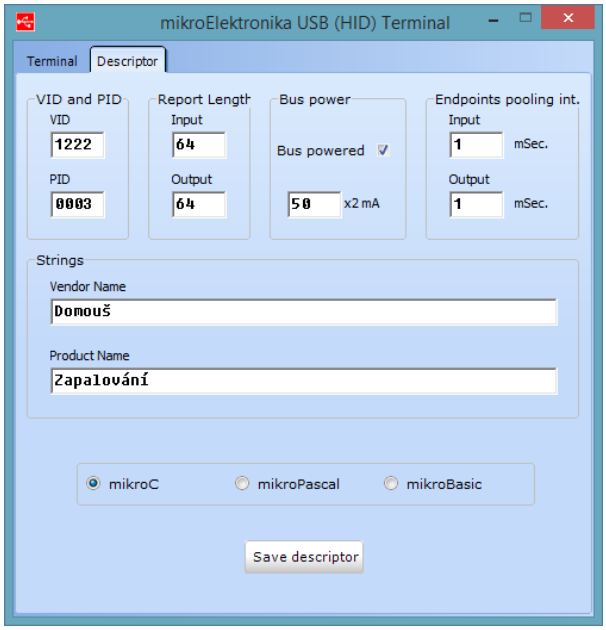
AN575 (uC)

8b		23b
Exp	|S|	Mantisa
31 24	23	22 0

Obr. 4.5 Porovnání typu float v PC a uC

Např. číslo -5 je v PC uloženo jako C0 A0 00 00, zatímco v uC jako 81 A0 00 00. Data je tedy potřeba přepočítat než je možné jejich uložení a použití v uC.

Dalším krokem při spouštění uC je povolení USB komunikace. To se provede voláním funkce `void HID_Enable(char *readbuff, char *writebuff);`.
Aby bylo volání této funkce možné je třeba do projektu soubor USBdsc.c, který obsahuje definice komunikace, jako jsou identifikační čísla VID (vendor ID) a PID (produkt ID), či název a popis zařízení. Tento soubor je možné vygenerovat automaticky pomocí HID terminálu přímo ve vývojovém prostředí (Obr. 4.6).



Obr. 4.6 HID Terminal vývojového prostředí mikroC

Nastavení uC:

Nejprve zakážeme analogové komparátory. To se provede odmaskováním registru CMCON hodnotou 0x07. Následuje nastavení všech analogových portů s výjimkou vstupu RA0 (tj. AN0) na digitální, vložením hodnoty 0x0E do registru ADCON1. Kanál AN0 bude použit k měření napětí baterie pro její ochranu před vybitím.
Dále nastavíme PORTB na kterém jsou vstupy a výstupy. Konkrétně RB0 na vstup a RB1 a RB4 na výstup.
TRISB = 0b000000001
Vhodně nastavíme čítače TMR0 a TMR1.

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = Timer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0**: Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

Obr. 4.7 Registr T0CON - převzato z ***[4]***

TMR0 nastavíme dle dříve zmíněných vlastností jako 16 bitový časovač s předděličkou 1:16 (16x) =>

T0CON = 0b00000011;

Zatím jej ponecháme vypnutý a spustíme jej až při prvním impulsu z čidla, aby nedocházelo k počítání falešných hodnot.

REGISTER 12-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of TImer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

bit 6 **T1RUN:** Timer1 System Clock Status bit
1 = Device clock is derived from Timer1 oscillator
0 = Device clock is derived from another source

bit 5-4 **T1CKPS1:T1CKPS0:** Timer1 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled
0 = Timer1 oscillator is shut off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T13CKI (on the rising edge)
0 = Internal clock (Fosc/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Obr. 4.8 Registr T1CON - převzato z ***[4]***

Čítač TMR1 nastavíme jako časovač (zdroj signálu je Fosc s 1:4 předděličkou a práci ve dvou 8b operacích.

T1CON = 0b00100000;

I tento čítač necháme zatím vypnutý.

Pokračujeme nastavením IPEN bitu z registru RCON na 1 a povolíme potřebná přerušení nastavením registru INTCON.

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

Obr. 4.9 Registr INTCON - převzato z ***[4]***

Povolíme přerušení s nízkou i vysokou prioritou (GIEH, GIEL) a povolíme přerušení od TMR0 a přerušení způsobené hranou na RB0 (tj. INT0).

INTCON = 0b11110000;

Ještě je potřeba povolit přerušení od čítače TMR1. K němu se přistupuje přes registr INTCON2, lej jednodušší způsob je nastavit přímo potřebný bit.

TMR0IE_bit = 1;

Výstupní port řídící zapalovací cívku nastavíme na úroveň log H, aby nedocházelo k vybíjení baterie a zahřívání cívky.

Nyní máme nastavený mikrokontrolér. Pustíme tedy program do nekonečné smyčky, ve které bude probíhat obsluha USB. Zbytek programu již probíhá v přerušeních.

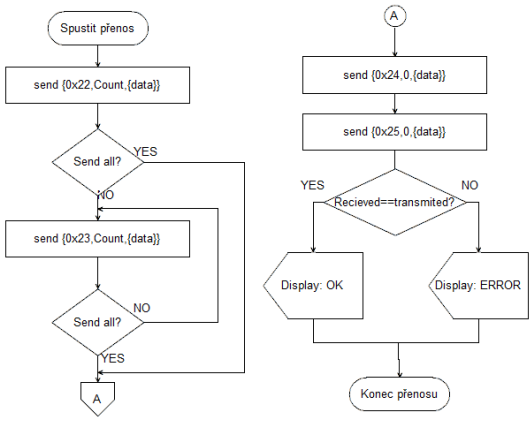
USB:

Komunikace probíhá pomocí dvou 64B bufferů *readbuff* a *writebuff*. První Byte definuje příkaz, ostatní jsou příslušná data. Procesor nikdy nezačíná komunikaci. Vždy čeká na příkaz z PC a reaguje na něj. K vyhodnocování příkazu je použit stavový automat.

Ve smyčce je nejprve obslouženo USB pomocí knihovní funkce *void USB_Polling_Proc(void);* Následně se funkcí *char HID_Read(void);* zkontroluje, jestli jsou k dispozici data a na konec se příchozí data vyhodnotí stavovým automatem a zpracují.

USB příkazy:

- 0x22
  - Začínám přenášet křivku předstihu.
  - Vynuluj čítače dat a přijmi první část křivky.
- 0x23
  - Odesílám další část křivky.
- 0x24
  - Odesílám poslední část křivky.
  - Přijmi zbylá data.
  - Přepočítej floaty z IEEE754 na AN575.
  - Ulož data do EEPROM.
- 0x25
  - Načti data z EEPROM
  - Přepočítej floaty z AN575 na IEEE754.
  - Odešli mi data zpět.
- 0x0F
  - Přejdi do bootloader módu (slouží k aktualizaci firmware)



Obr. 4.10 Blokový diagram SW pro nahrávání křivky

Přerušení:

Pokud dojde v uC k události, která je spojena s odmaskovaným (povoleným) přerušením, dojde k zavolání funkce void interrupt (){ … }.
Tato funkce obsahuje veškeré obsluhy přerušení. Jaké přerušení má být vykonáno rozhodují až podmínky, reagující na flag bit daného přerušení, v oné funkci.

Jednotlivá přerušení:

- TMR0 (TMR0IF)
    - Vypne čítač TMR0
    - Nastaví řídící výstup na log 1 (přestane nabíjet cívku).
- TMR1 (TMR1IF)
    - Příznak *zapal* je nastaven
        - Nastaví řídící výstup na log 1 (hodí jiskru)
        - Nastaví čítač TMR1 na minulou hodnotu
        - Vynuluje příznak *zapal*
    - Příznak *zapal* je vynulován
        - Vynuluje řídící výstup (začne nabíjet cívku)
        - Vypne čítač TMR1
- INT0 (INT0IF)
    - Čítač TMR0 je zapnutý
        - Načte hodnotu čítače TMR0
        - Vynuluje TMR0
        - Spočítá hodnotu pro TMR1 dle rovnice

$$TMR1 = 2^{16} - TMR0 + 18 + krzpoz$$

        - Spustí čítač TMR0
        - Určí aktuální křivku dle hodnoty TMR0 a pole DOTMR
        - Spočítá int krzpoz dle rovnice

$$krzpoz = k[aktkrivka] * TMR0 + q[aktkrivka]$$

    - Čítač TMR0 je vypnutý
        - Vynuluje čítač TMR0
        - Spustí TMR0

Bootloader:

Bootloader je malý program umístěný na konci paměti, který slouží k aktualizaci firmwaru zapalování bez nutnosti připojení programátoru. Jeho hlavní funkce *void bootloader()* je umístěna na pevně daném místě v paměti (v mém případě na adrese 0x7D50). V programu, který má být nahráván stačí přidat definici funkce bootloader() a z vhodného místa ji zavolat. Před zavoláním funkce je nejprve nutné ukončit případné USB připojení pomocí zavolání funkce *void HID_Disable(void).* Pokud by funkce bootloader nebyla volána, bylo by sice nahrání programu možné, ale po restartování uC bychom se do bootloaderu znovu nedostali, další přeprogramování by tedy nebylo možné

Definice k přidání do programu:

```
#pragma funcorg bootloader 0x7D50
void bootloader(){
}
```

Bootloader přepisuje původní program pomocí knihovní funkce
void FLASH_Erase_Write_64(long address, char data);*
Komunikace probíhá pomocí USB. První Byte slouží k identifikaci příkazu. Pokud obsahuje hodnotu 0x0F jsou data použity bootloaderem, obsahuje-li jinou hodnotu, jsou data ignorována. Druhý byte je považován za příkaz. Příkazy jsou vyhodnocovány pomocí stavového automatu. Každý příkaz po svém spuštění odešle odezvu (ACK) ve formátu {0x0f, cmd}
Příkazy bootloaderu:

- 0x00 – cmdNON — Neprovádět žádnou akci.
- 0x01 – cmdSYNC — Synchronizovat s PC (není funkční)
- 0x02 – cmdINFO — Odeslat informace (verzi FW, verzi bootloaderu,…)
- 0x03 – cmdBOOT — Přejít do bootloaderu
- 0x04 – cmdREBOOT — Restartovat uC
- 0x11 – cmdWRITE — Zapsat program do FLASH paměti
- 0x12 – cmdREAD — Přečíst program z FLASH (není funkční)
- 0x21 – cmdERASE — Odstranit program z uC (není funkční)

Blíže se podíváme na jednotlivé příkazy, které již byly zprovozněny.

cmdBOOT – Slouží k přechodu do bootloaderu. Pokud v něm již jsme, pouze vrátí ACK.

cmdREBOOT – volá příkaz reset, který bohužel není v kompileru zaveden. Je tedy nutno použít direktivu ASM pro zadání části kódu přímo v asembleru. *asm RESET;*

cmdWRITE – Přijímá pouze data reprezentující počáteční adresu a délku programu, která bude odeslána. Následně nastaví příznak, odešle ACK a čeká na příchod dat, která nyní již neobsahují žádnou hlavičku (Jedná se přímo o data k uložení do paměti FLASH)

Příklad komunikace při cmdWRITE:

PC => uC

STX	CMD	ADRESS				COUNT	
0x0F	0x11	0x00	0x00	0x00	0x00	0x00	0x80

uC=>PC

STX	CMD
0x0F	0x11

PC=>uC

Data[0]	Data[1]	Data[2]						Data[62]	Data[63]

uC=>PC

STX	CMD
0x0F	0x11

PC=>uC

Data[64]	Data[65]	Data[66]						Data[126]	Data[127]

uC=>PC

STX	CMD
0x0F	0x11

Data z příkladu budou uložena v paměti FLASH na adrese 0 – 127.

Tento postup platí i při rozsáhlejších programech, pouze neproběhne pouze dvakrát, ale vícekrát vždy po 64B.

Příkaz cmdWRITE umí přepsat i paměť EEPROM, CONFIG a UID mikrokontroléru. K tomu je potřeba odeslat data stejným způsobem jako při nahrávání programu, pouze s vysokou adresou.
Konkrétně:
0x400000 pro EEPROM
0x300000 pro CONFIG
0x200000 pro USER ID

Z použití tohoto postupu při rozlišování typu dat vyplívá omezení maximální velikosti FLASH paměti mikrokontroléru na 2MB

Bootloader zároveň obsahuje ochranu proti přepsání sama sebe. Pokud je adresa programu přesáhne adresu definovanou v bootloaderu jako bootloader_start a nejedná se o speciální adresy zmíněné výše, bootloader data pouze potvrdí pomocí ACK, ale nikam je nezapisuje.

Bootloader je zcela samostatný program, který je jako jediný do uC nahraný pomocí programátoru. Samotný program pro zapalování je do uC nahrán přes bootloader.

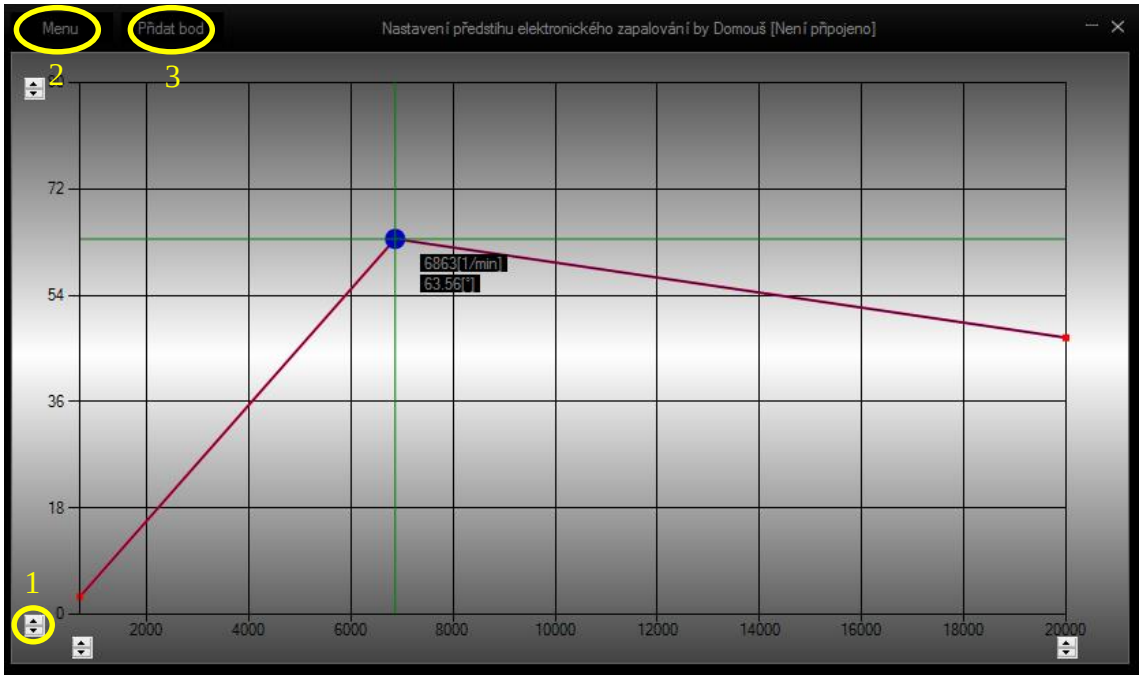
PC:

Počítačová aplikace pro vytváření a nahrávání křivek je napsána v jazyce C# jako „Windows Form Aplication“. Program je zde vytvářen přetahováním předpřipravených prvků do formuláře. K takto připravené aplikaci se potom připíše funkce pro jednotlivé prvky.

Program je rozdělen do několika částí:

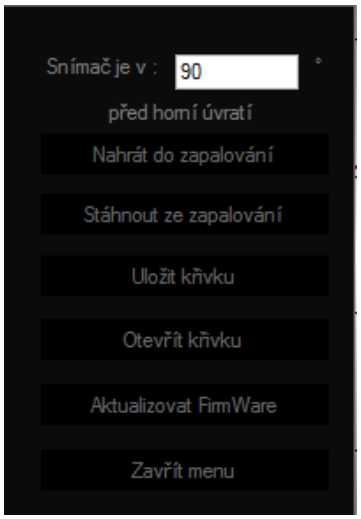
- Program.cs
    - Obsahuje hlavní funkci programu `static void Main (){},` která je volána po spuštění programu.
    - Vytváří a zobrazuje instanci formuláře.
- Form.Designer.cs
    - Obsahuje automaticky generovaný kód, který popisuje vzhled formuláře.
- Form.cs
    - Obsahuje jednotlivé metody reagující na akce z formuláře (změna hodnoty, najetí myší, kliknutí, zmáčknutí klávesy,…)

Hlavní částí programu je graf (prvek chart), ve kterém je zobrazena křivka a je možné ji upravovat (přidávat, odebírat, přetahovat body).

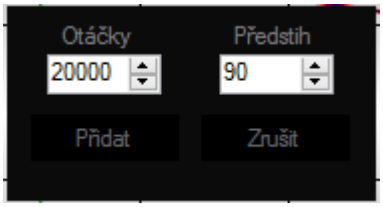


Obr. 4.11 Náhled finální aplikace pro správu křivek (modrý bod je vybraný)

Dále jsou zde 4 prvky typu NumericUpDown (Obr. 4.11 – 1) sloužící na nastavení rozsahů osy. Tlačítko Menu (2) pro zobrazení menu (Obr. 4.12) a tlačítko Přidat bod (3) které zobrazí prvky pro přidání bodu na přesné místo (Obr. 4.12). V titulku okna je navíc zobrazeno, zda je zapalování připojeno do USB.



Obr. 4.12 Hlavní menu aplikace



Obr. 4.13 Menu pro přidání bodu

Ovládání aplikace:

- Pro základní práci s křivkou je možné použít přímo hlavní graf.
  - Jedním kliknutím do grafu je vybrán nejbližší bod.
  - Přetahováním kurzoru se stisknutým tlačítkem myši je přesouván vybraný bod
  - Tlačítky W a S je možné měnit předstih vybraného bodu o +- 0.1°
  - Tlačítky A a D je možné měnit otáčky vybraného bodu o +- 10 1/min
  - Tlačítkem BackSpace je možné odstranit vybraný bod
- Další úkony je možné provádět z menu.
  - Uložit aktuální křivku do počítače.
  - Otevřít uloženou křivku
  - Nahrát křivku do zapalování
  - Stáhnout křivku ze zapalování
  - Nastavit umístění snímače
  - Otevřít okno pro aktualizaci FW (bootloader) (Obr. 4.16, Obr. 4.17)

USB:

Ke komunikaci pomocí USB je použita knihovna UsbHid, kterou napsal **Szymon Roslowski** a je dostupná pod licencí **CPOL (The Code Project Open License).** na adrese http://www.codeproject.com/Tips/530836/Csharp-USB-HID-Interface.
K využívání knihovny je potřeba ji přidat do projektu a mezi závislosti našeho programu. Dále je potřeba vytvořit instanci zařízení a delegáta na invokování příchozích dat.

```
public UsbHidDevice Device;
delegate void SetTextCallback(byte[] InputData);
```

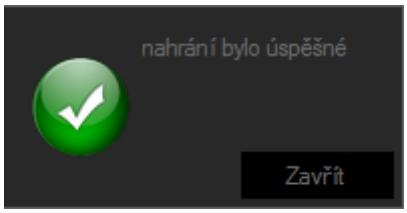
Nyní je potřeba definovat, o jaké zařízení se jedná a jaké funkce budou reagovat na zařízení (Připojení, odpojení, příjem dat,…). To se provede po načtení aplikace.

```
Device = new UsbHidDevice(0x1222, 0x0003);
Device.DataReceived += DeviceDataReceived;
Device.OnDisConnected += pripoj;
pripoj();
```

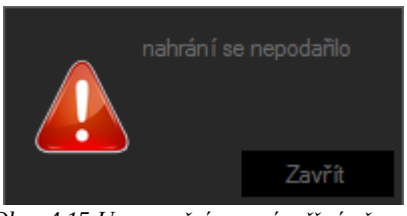
Příchozí data pocházejí z jiného vlákna. Je tedy potřeba je nejprve invokovat.

```
if (this.InvokeRequired)
{
    SetTextCallback d = new SetTextCallback(SetData);
    this.Invoke(d, new object[] { InputData });
}
```

Program dále komunikuje podle výše zmíněných specifikací. Postupně odešle celou křivku a nechá si je poslat zpátky (Obr 4.10). Porovná odeslaná a přijatá data a zobrazí příslušnou zprávu.



Obr. 4.14 Upozornění na úspěšný přenos



Obr. 4.15 Upozornění na neúspěšný přenos

Popis metod programu:

- Konstruktor `Form1`
    - Vykreslí formulář zavoláním metody `InitializeComponent();`
    - Změní kurzor tlačítek exit a minimalize na ruku
    - Přidá body do křivky na maximální a minimální otáčky s předstihem 8°
    - Zavolá metodu prepocitej();
- Metoda `Form1MainLoad` – reaguje na načtení formuláře
    - Nastaví instanci USB zařízení
    - Zavolá metodu připoj()
- Metoda `pripoj`
    - Zkusí připojit zařízení
    - Změní titulek okna (Připojeno/Není připojeno)
    - Povolí/zakáže časovač connect
- Metoda `DeviceDataReceived`
    - Invokuje data
    - Zavolá metodu SetData
- Metoda `SetData`
    - Přijme data
    - Porovná odeslaná a přijatá data
    - Zobrazí správu o úspěšnosti přenosu
    - Načte data a převede je na křivku (možnost stáhnout ze zapalování)
- Metoda `Z_EEPR_NA_BODY`
    - Přepočítá hodnoty DOTMR k a q na křivku předstihu v závislosti na otáčkách
    - Zobrazí načtenou křivku
    - Vypíše upozornění „`Křivka byla načtena`“
- Metoda `close_Click` – reaguje na kliknutí na tlačítko close
    - Zavře aplikaci
- Metoda `minimalize_Click` – reaguje na kliknutí na tlačítko minimalizuj
    - Minimalizuje aplikaci
- Metoda `Form1_MouseDown` – reaguje na stisk tlačítka myši
    - Zahájí přesouvání okna aplikace

- Metoda `Form1_MouseUp` – reaguje na uvolnění tlačítka myši
    - Ukončí přetahování okna
- Metoda `Form1_MouseMove` – reaguje na změnu polohy myši
    - Změní polohu okna aplikace
- Metoda `chart1_DoubleClick` – reaguje na dvojklik myší do grafu
    - Přidá do křivky předstihu bod
- Metoda `chart1_MouseDown` - reaguje na stisk tlačítka myši v grafu
    - Započne přetahování vybraného bodu
- Metoda `Compare1`
    - Porovná dva body (KeyValuePair)
- Metoda `chart1_MouseUp` - reaguje na uvolnění tlačítka myši v grafu
    - Ukončí přetahování bodu
- Metoda `chart1_MouseMove` - reaguje na změnu polohy kurzoru myši v grafu
    - Změní polohu bodu
- Metoda `chart1_MouseEnter` – reaguje na najetí kurzoru myši nad graf
    - Skryje kurzor
- Metoda `chart1_MouseLeave` – reaguje na vyjetí kurzoru myši z grafu
    - Zobrazí kurzor
- Metoda `chart1_KeyPress` – reaguje na stisk libovolné klávesy
    - W – zvedne předstih bodu o 0.1°
    - S – sníží předstih bodu o 0.1°
    - A – sníží otáčky bodu o 10 1/min
    - D – zvedne otáčky bodu o 10 1/min
    - BackSpace – odstraní bod
    - Enter – zobrazí dialog pro přidání bodu
- Metoda `Polyfit`
    - Proloží vstupní body polygonem zadaného řádu
- Metoda `prepocitej`
    - Proloží body
    - Přepočítá proložení na hodnoty čítače
    - Uloží dílčí křivky do pole
- Metoda `button1_Click` – reaguje na stisk tlačítka „Menu“

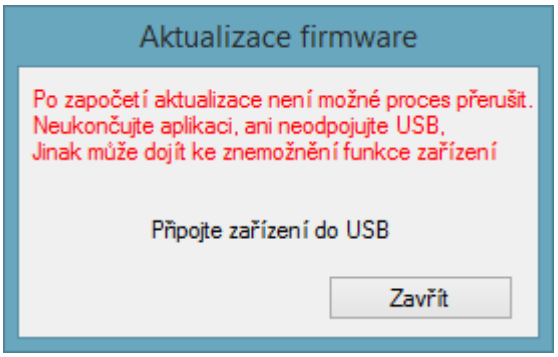
    - Zobrazí hlavní menu
- Metoda `button5_Click` - reaguje na stisk tlačítka „Zavřít menu“
    - Skryje hlavní menu
- Metoda `button3_Click` - reaguje na stisk tlačítka „Uložit křivku“
    - Zobrazí dialog pro uložení křivky (SaveFileDialog)
    - Může zobrazit upozornění „Probíhá jiná operace“
- Metoda `button6_Click` – reaguje na stisk tlačítka „Otevřít křivku“
    - Zobrazí dialog pro načtení křivky (OpenFileDialog)
    - Může zobrazit upozornění „Probíhá jiná operace“
- Metoda `saveFileDialog1_FileOk` – reaguje na potvrzení dialogu
    - Uloží křivku do souboru
- Metoda `openFileDialog1_FileOk` – reaguje na potvrzení dialogu
    - Načte křivku ze souboru
- Metoda `button2_Click` – reaguje na kliknutí tlačítka „Nahrát do zapalování“
    - Odešle křivku do zapalování po 54B
- Metoda `Compare`
    - Porovná dvě pole Bytů
- Metoda `numericUpDown1_ValueChanged` až `numericUpDown4_ValueChanged` -
    - Reaguje na změnu hodnoty NumericUpDown
    - Změní rozsah os grafu
- Metoda `connect_Tick` – reaguje na uplynutí časovače
    - Zavolá metodu pripoj
- Metoda `button4_Click` – reaguje na kliknutí tlačítka „Stáhnout ze zapalování“
    - Odešle příkaz pro navrácení dat ze zapalování (0x25)
- Metoda `button7_Click` – reaguje na kliknutí tlačítka „Zavřít“
    - Skryje upozornění a zobrazí menu
- Metoda `textBox5_TextChanged` – reaguje na změnu hodnoty umístění snímače
    - Aktualizuje umístění snímače, rozsahy os a přepočítá
- Metoda `button10_Click` – reaguje na kliknutí tlačítka „Přidat bod“
    - Zobrazí dialog pro přidání bodu
- Metoda `button12_Click` - reaguje na kliknutí tlačítka „Zavřít“
    - Skryje dialog pro přidání bodu

- Metoda `button11_Click` - reaguje na kliknutí tlačítka „Přidat“
    - Přidá bod na zadanou pozici
- Metoda `button5_Click_1` – reaguje na kliknutí tlačítka „Aktualizovat FW“
    - Zobrazí okno bootloaderu
- Metoda `attention`
    - Zobrazí upozornění odeslané v argumentu

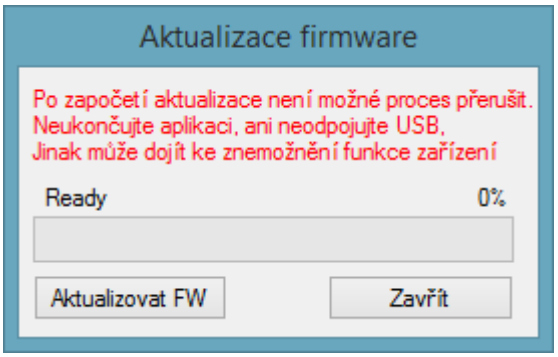
Bootloader:

Bootloader je samostatné okno aplikace, které je vytvořeno a zobrazeno původní aplikací, která je při zobrazení bootloaderu skryta a opětovně zobrazena při zavření bootloaderu.

Bootloader má dvě možnosti zobrazení v závislosti na přítomnosti USB zařízení.



Obr. 4.16 Okno bootloaderu bez připojení



Obr. 4.17 Okno bootloaderu s připojením

Po připojení USB se zobrazí tlačítko „Aktualizovat FW“. Kliknutím na toto tlačítko vyvoláme dialog pro výběr „.hex“ souboru. Po potvrzení dialogu je již postup automatický. Bootloader nejprve načte soubor a rozkóduje jej z formátu zmíněného níže. Rozkódovaná data uloží do čtyř polí (pro program, config, uid a eeprom) a doplní volná místa. Restartuje mikrokontrolér do bootloader módu postupně odešle všechna pole s příslušnou adresou pro uC verzi bootloaderu (viz. bootloader verze pro uC). Po nahrání všech dat opět restartuje uC a zobrazí potvrzení úspěšné aktualizace. Po celou dobu je v bootloaderu zobrazeno upozornění na odpojení zařízení od PC.

Popis „.hex“ souboru:

Mikrokontroléry od firmy Microchip využívají souborů **extended Intel Hex** (Přesněji INHX32)

Soubor se skládá z linek začínajících „:“. Každý řádek udává část kódu s určitou adresou a délkou.

Linky mají formát:
:BBAAAATT[DDDDDDDD]CC

BB – Počet Bytů na daném řádku

AAAA – Adresa začátku řádku v Bytech

EE – Typ dat

- 00 znamená data
- 01 znamená End Of File
- 04 znamená posun adresy

DD – Data

CC – Checksum

Jednotlivé typy dat jsou odděleny pomocí linky pro posuv adresy.

Před programem je umístěn řádek „:020000040000FA“

Před daty pro EEPROM je umístěn řádek „:0200000400F00A“

Před konfigurací je umístěn řádek „:020000040030CA“

Před User ID je řádek „:020000040020DA“

Konec souboru je označen řádkem „:00000001FF“

Příklad:

```
:100000003C932085BBE0697A342C4D261FB267A4F2
:100030005216E94D424E627679C2337EA57C2895157
:0200000400F00A
:10000000FFFFAAFFFFFBBFF54FFFFF01FFFFFFFF00
:020000040030CA
:1000000000270F0F0037F50125797160FF400000C5
:020000040020DA
:08000000010203040506070 8D4
:00000001FF
```

Příklad obsahující dva řádky programu, 1 řádek EEPROM, 1 konfigurace a 1 UID.

Zkouška zapalování

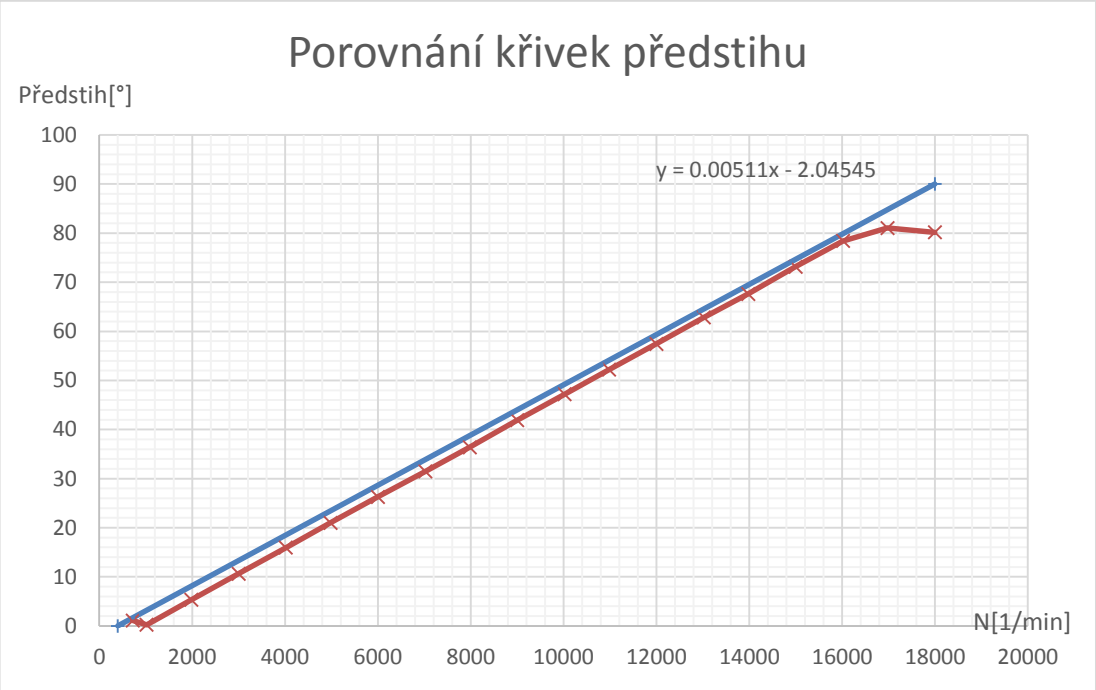
Nyní když máme napsané všechny programy, můžeme začít s testování.

Zkouška s generátorem obdélníkových pulzu:

Nejprve je vhodné vyzkoušet program na osciloskopu, nebo logickém analyzéru za pomoci frekvenčního generátoru. Vytvoříme tedy nějakou jednoduchou křivku (pro začátek například lineárně stoupající předstih) a nahrajeme ji do zapalování. Potom budeme postupně zvedat frekvenci na generátoru a ze zpoždění signálu počítat předstih.

Z poměru doby zpoždění a periody vynásobené 360 se spočítá úhel zpoždění, který uběhne od čidla. Díky znalosti pozice čidla můžeme spočítat předstih.

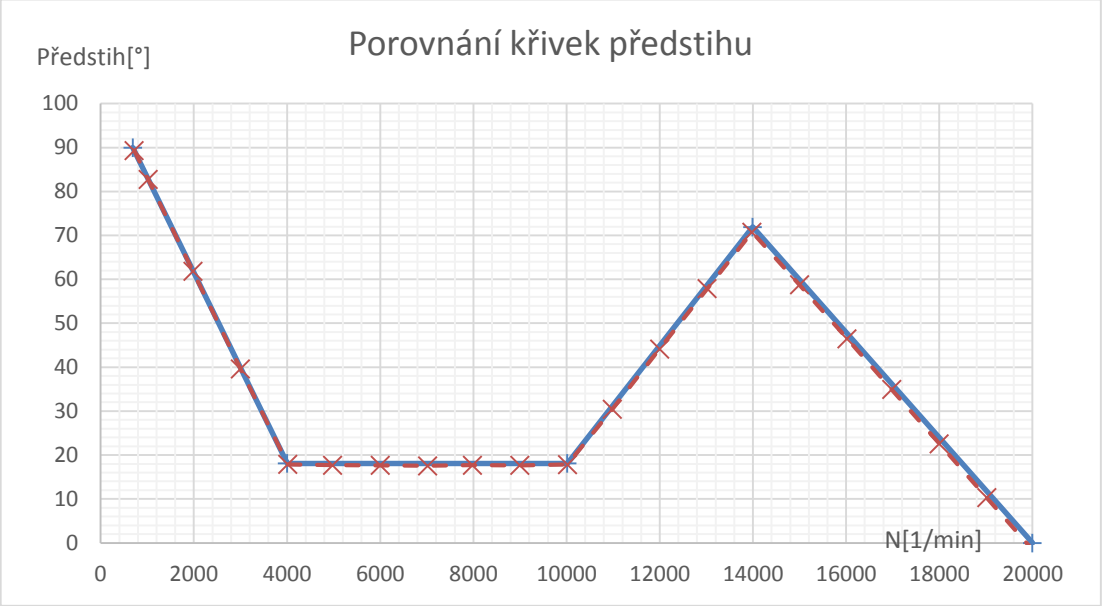
$$\varphi_{preds} = 90 - \frac{T_{zpoz}}{T} * 360 = 90 - \frac{N * T_{zpoz}}{60} * 360$$



Obr. 4.18 Kontrola jednoduché křivky předstihu

Jak je vidět změřená křivka předstihu je velmi podobná se zadanou. Absolutní chyba vypadá téměř konstantní v celém rozsahu, kromě jeho konce. Dala by se tedy vylepšit posunutím žádané křivky.

Dalším krokem je vyzkoušení složitější křivky. Nahrajeme tedy křivku s několika zlomy, stoupáním, klesáním a lineární částí. Potom provedeme měření znovu.



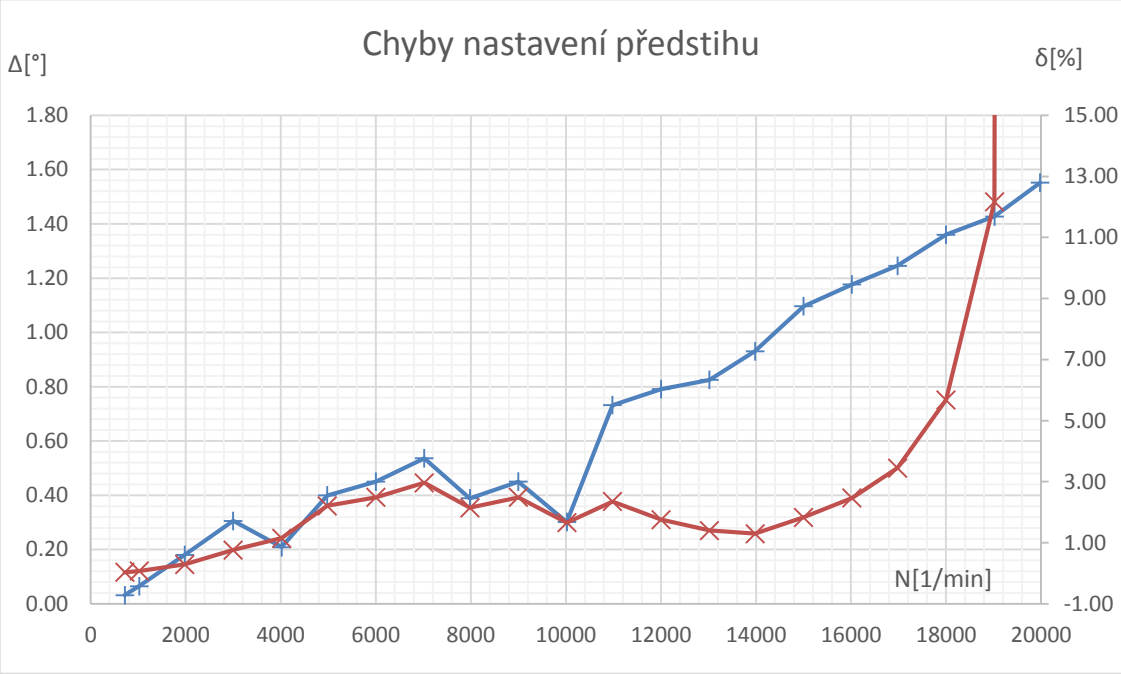
Obr. 4.19 Kontrola složitější křivky předstihu

V tomto měření vypadají křivky téměř totožně. Přikládám tedy ještě tabulku naměřených hodnot a vypočítaných chyb.

N[1/min]	T_{zp}	f[Hz]	T[ms]	φ_{zpoz}[°]	φ_{predsM}[°]	φ_{predsZ}[°]	Δ_{preds}[°]	δ_{preds}[%]
720	0.158	12	83.33	0.68	89.32	89.35	0.03	0.04
1020	1.18	17	58.82	7.22	82.78	82.84	0.07	0.08
1980	2.37	33	30.30	28.16	61.84	62.03	0.18	0.29
3000	2.8	50	20.00	50.40	39.60	39.91	0.31	0.77
4020	2.99	67	14.93	72.12	17.88	18.09	0.21	1.15
4980	2.42	83	12.05	72.31	17.69	18.09	0.40	2.21
6000	2.01	100	10.00	72.36	17.64	18.09	0.45	2.49
7020	1.72	117	8.55	72.45	17.55	18.09	0.54	2.97
7980	1.51	133	7.52	72.30	17.70	18.09	0.39	2.15
9000	1.34	150	6.67	72.36	17.64	18.09	0.45	2.49
10020	1.2	167	5.99	72.14	17.86	18.16	0.30	1.66
10980	0.9046	183	5.46	59.60	30.40	31.14	0.73	2.35
12000	0.637	200	5.00	45.86	44.14	44.93	0.79	1.76
13020	0.411	217	4.61	32.11	57.89	58.72	0.83	1.41
13980	0.2293	233	4.29	19.23	70.77	71.70	0.93	1.30
15000	0.347	250	4.00	31.23	58.77	59.87	1.10	1.83
16020	0.4528	267	3.75	43.52	46.48	47.65	1.18	2.47
16980	0.5407	283	3.53	55.09	34.91	36.16	1.25	3.45
18000	0.6242	300	3.33	67.41	22.59	23.95	1.36	5.68
19020	0.69833	317	3.15	79.69	10.31	11.73	1.43	12.16
19980	0.7617	333	3.00	91.31	-1.31	0.24	1.55	648.13

Tab. 4.1 Naměřené hodnoty a chyby zapalování

I zde je veliký rozdíl pouze na konci rozsahu. Je zde vidět, že chyba je vždy na konci rozsahu, bez závislosti na maximálních otáčkách. Můžeme si zobrazit závislost chyby na otáčkách, ale bude vhodné vynechat poslední hodnotu, či dvě.



Obr. 4.20 Absolutní a relativní chyba zadaného předstihu.

Z této charakteristiky je vidět, že absolutní chyba nepřesáhne 1.6°. Tato chyba je multiplikativní a bylo by ji možné částečně vykompenzovat podělením směrnicí proložení absolutní chyby.

Zkouška v motoru:

Nyní když jsme zjistili, že zapalování výborně počítá a nastavuje předstih, nastal čas vyzkoušet jej přímo v motoru. Připojíme zapalování k motoru, jak bylo zmíněno výše (viz. 4. Realizace zapalování). Nahrajeme křivku s konstantním předstihem odpovídajícím původnímu předstihu motoru a pokusíme se motor nastartovat. Motor normálně nastartoval, ale když jsem na motor přidělal úhloměr a pokusil se změřit předstih stroboskopem (který jsem sám postavil), viděl jsem, jak předstih kmitá okolo 70° místo zadaných 20. Po dalším měření jsem zjistil, že problém je v kmitajících volnoběžných otáčkách. Jelikož se doba otáčky výrazně mění mezi otáčkami a výpočet pracuje s daty z předchozí otáčky, dochází k chybě předstihu. Sice se jedná o jev probíhající pouze při volnoběhu, ale i tak by to mohlo způsobovat problémy.

Naměřil a porovnal jsem pár hodnot po sobě jdoucích otáček a dal je do následující tabulky:

				Předstih	
t[ms]	f[Hz]	N[1/min]	zp	Žádaný	Reálný
17.18	58.21	3492.43	4.30	90	
17.176	58.22	3493.25	4.29	90	90.02
18.88	52.97	3177.97	4.72	90	81.88
18.44	54.23	3253.80	4.61	90	92.15
16.587	60.29	3617.29	4.15	90	100.05
16.1086	62.08	3724.72	4.03	90	92.67
16.755	59.68	3581.02	4.19	90	86.53
17.255	57.95	3477.25	4.31	90	87.39
17.89	55.90	3353.83	4.47	90	86.81
18.53	53.97	3237.99	4.63	90	86.89
16.47	60.72	3642.99	4.12	90	101.26

Tab. 4.2 Porovnání po sobě jdoucích otáček motoru.

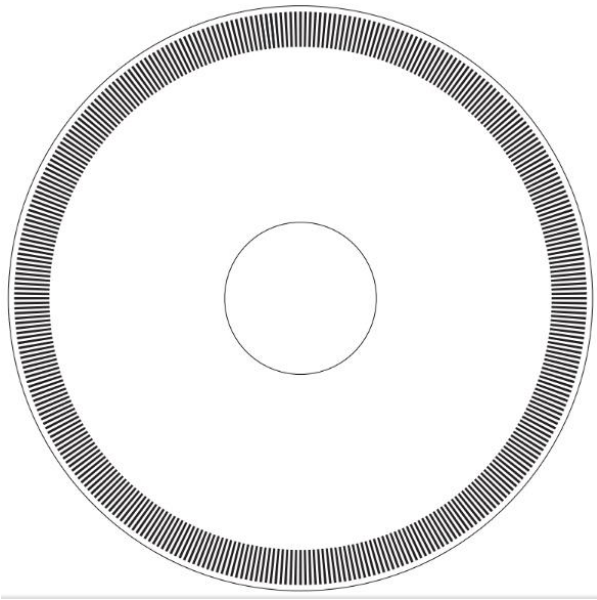
Jak je vidět, při žádaném předstihu 90° se skutečný předstih bude pohybovat mezi 81° a 101°. Pokud by byl tedy předstih žádaných 22° mohl by se pohybovat mezi 13°a 33°.

Jelikož mě nenapadlo, jak tuto chybu vykompenzovat se současnou konfigurací, jsem nucen ji změnit.

Nabízejí se dvě možnosti. První možností je přidat druhý snímač (reakci na druhou hranu snímače), např. do polohy 180° a počítat přímo s aktuálními daty. Tato možnost by byla reálná, ale současný setrvačník by se špatně upravoval a přesnost by se nemusela příliš zvýšit.

Druhou možností je přidat do motoru inkrementální enkodér. Tato možnost se sice zdá být komplikovanější, ale ve skutečnosti stačí přidat kódovací disk (Obr 4.21), který bude přidělaný k setrvačníku a optickou závoru.

Kódovací disk jsem nechal vysvítit na fotoplotru s přesností 1200dpi a 360 impulzy na otáčku a přilepil jej na disk z polykarbonátu. Hallův snímač jsem v motoru ponechal, aby nám udával absolutní natočení motoru.



Obr. 4.21 Kódovací disk

Po přidání enkodéru již není problém upravit programy. Do programu pro uC se přidá jedno přerušení, které bude přičítat polohu natočení a reagovat na výstup z optické závory. V přerušení, které reaguje na hallovu sondu umažeme nastavení čítače a hodnotu krzpoz převedeme do proměnné udávající úhel zapálení. Nyní stačí upravit počítačovou aplikaci tak, aby neposílala hodnoty pro čítač, ale přímo předstih ve stupních. Po těchto úpravách je možné přehrát firmware pomocí bootloaderu a přistoupit k dalším testům.

Jelikož můj generátor má pouze jeden výstup, zvolil jsem jako nejlepší postup připevnit setrvačník do vrtačky a otestovat zapalování pouze s pomocí osciloskopu. Při prvním pokusu jsem k setrvačníku připojil skutečný inkrementální enkodér s 600 impulzy na otáčku místo kódovacího disku. Do zapalování jsem nejprve nahrál křivku s konstantním předstihem přibližně 22° a později křivku s lineárně stoupajícím předstihem. Obě křivky jsem proměřil pomocí osciloskopu a stroboskopem. Data jsem uložil do následujících dvou tabulek a grafů. Měření stroboskopem není zcela přesné, ale ukázalo přibližně očekávaný průběh. To potvrdilo i měření na osciloskopu. Chyba měření může být dána nepřesným měřením času pomocí kurzorů, nebo chybou zaokrouhlování v uC.

Naměřené hodnoty křivky s konstantním předstihem 22°:

N[1/min]	f[Hz]	T[ms]	T_{zp}[ms]	φ_{zpoz}[°]	φ_{preds}[°]	Δ_{φ}[°]	δ_{φ}[%]
822.26	13.70	72.97	68.31	337.01	22.99	0.29	1.26
999.33	16.66	60.04	56.15	336.68	23.32	0.62	2.68
1222.74	20.38	49.07	45.88	336.60	23.40	0.70	3.01
1373.94	22.90	43.67	40.99	337.91	22.09	-0.61	-2.75
1625.58	27.09	36.91	34.47	336.20	23.80	1.10	4.62
1791.58	29.86	33.49	31.31	336.57	23.43	0.73	3.13
1951.85	32.53	30.74	28.82	337.51	22.49	-0.21	-0.95
2039.43	33.99	29.42	27.6	337.73	22.27	-0.43	-1.93
2225.52	37.09	26.96	25.31	337.97	22.03	-0.67	-3.03
2472.19	41.20	24.27	22.81	338.34	21.66	-1.04	-4.82

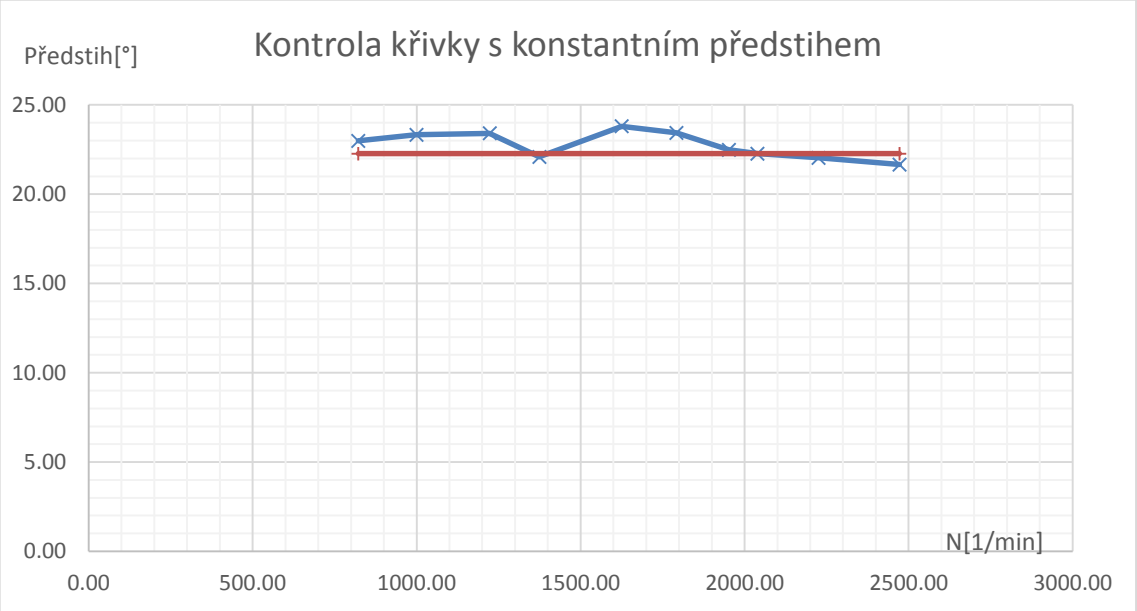
Tab. 4.3 Naměřené hodnoty s enkodérem a křivky s konstantním předstihem

Naměřené hodnoty křivky s lineárně stoupajícím předstihem:

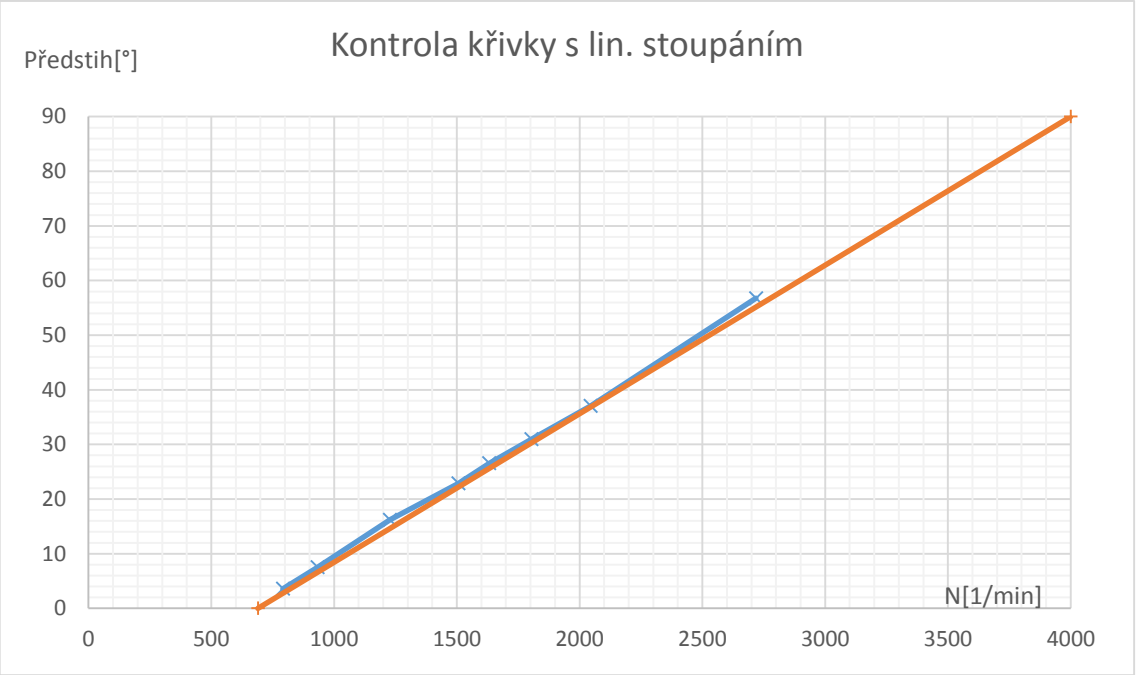
N[1/min]	f[Hz]	T[ms]	T_{zp}[ms]	φ_{zpoz}[°]	φ_{preds}[°]	φ_{pzad}[°]	Δ_{φ}[°]	δ_{φ}[%]
791.87	13.20	75.77	75.02	356.44	3.56	2.77	-0.79	-28.65
932.55	15.54	64.34	63.00	352.50	7.50	6.59	-0.90	-13.69
1227.50	20.46	48.88	46.68	343.80	16.20	14.61	-1.59	-10.87
1506.40	25.11	39.83	37.30	337.13	22.87	22.20	-0.67	-3.01
1632.21	27.20	36.76	34.05	333.46	26.54	25.62	-0.92	-3.59
1803.43	30.06	33.27	30.41	329.05	30.95	30.27	-0.67	-2.22
2044.29	34.07	29.35	26.33	322.96	37.04	36.82	-0.22	-0.59
2718.62	45.31	22.07	18.59	303.24	56.76	55.16	-1.61	-2.91

Tab. 4.4 Naměřené hodnoty s enkodérem a křivky s lineárně stoupajícím předstihem

Charakteristiky:



Obr. 4.22 Kontrola křivky s konstantním předstihem a enkodérem

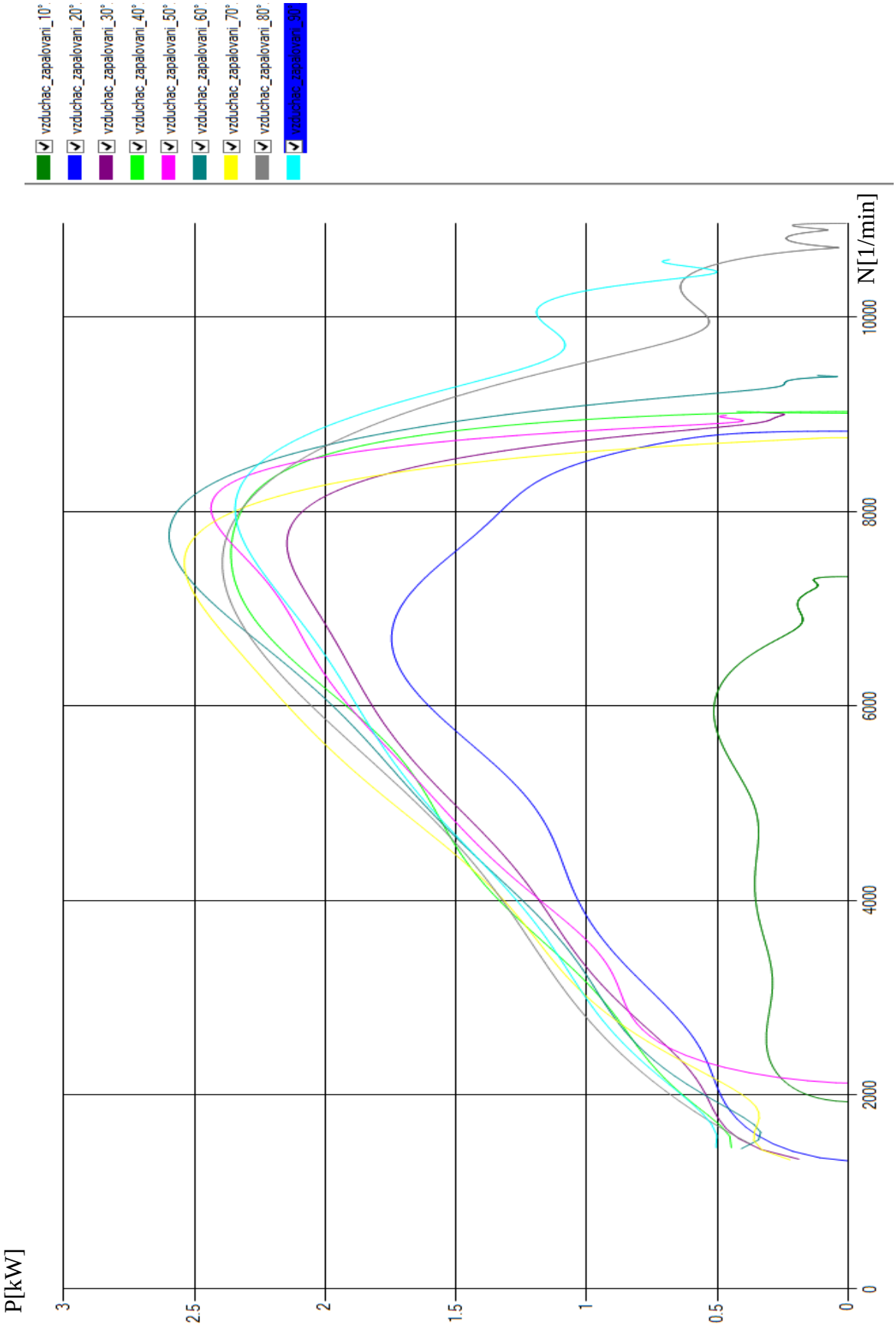


Obr. 4.23 Kontrola křivky s lineárně stoupajícím předstihem a enkodérem

Z naměřených charakteristik je vidět, že zapalování nastavuje chybu poměrně přesně a to s chybou do dvou stupňů. Relativní chyba vychází u konstantního předstihu do 5%. U lineárně stoupajícího vychází zpočátku větší a postupně se zmenšuje.

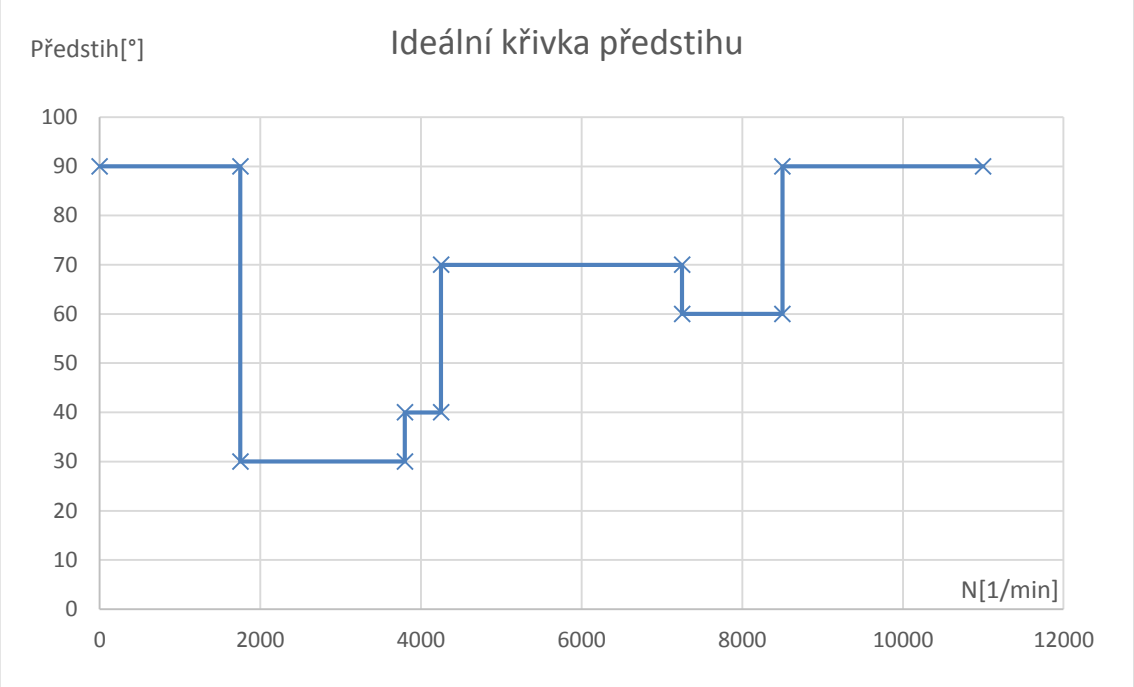
Nyní když máme ověřenou funkci zapalování, nastal čas jej otestovat v motoru. Bohužel jsem již neměl čas na implementaci enkodéru do motoru, kódovací disk jsem poškodil a nový mi nestihli včas vysvítit. Finální zkoušku s reálným motorem a inkrementálním enkodérem jsem bohužel nemohl provést, ale věřím, že by vše fungovalo na výbornou.

Jelikož jsem si myslel, že stihnu zapalování dokončit včas, postavil jsem ještě válcovou zkušebnu na měření výkonu a momentu motoru. Nemohl jsem sice změřit kompletní zapalování, ale alespoň jsem změřil postupně křivky výkonu pro předstihy od 0° do 90° po 10°. Jednotlivé křivky jsem vložil do jednoho grafu (Obr. 4.24) a jejich porovnáním jsem získal ideální křivku předstihu (Obr. 4.25). Tato křivka sice neodpovídá předpokladu (předstih stoupá s otáčkami), ale alespoň můžu po přidělání enkodéru z něčeho vycházet. Křivku nemohu zatím ani ověřit.



Obr. 4.24 Charakteristiky výkonu s různými předstihy

Nyní vybereme v každých otáčkách předstih s nejvyšším momentem a tím vytvoříme ideální křivku předstihu.



Obr. 4.25 Ideální křivka předstihu

Obr. 4.26 Ideální křivka předstihu získaná proložením

5. ZÁVĚR

Úspěšně jsem splnil zadání a navrhnul kompletní obvodové řešení elektronického tranzistorového zapalování. Kompletní zapojení jsem otestoval pomocí frekvenčního generátoru zapojené na nepájivém poli. Po pár komplikacích způsobených například kondenzátorem C7, který jsem během chvíle propálil vysokonapěťovými překmity vznikajícími při odpojení cívky, nebo vnitřní indukčností odporu R3, která způsobovala restartování uC jsem obvod úspěšně rozeběhnul. Stačilo vyměnit kondenzátor za fóliový do 630V a odpor nahradit SMD verzí. Dále jsem navrhnul desku plošných spojů a napsal potřebné softwarové vybavení pro mikrokontrolér i počítač. Dále jsem obě verze programu vybavil bootloaderem pro pozdější aktualizaci firmwaru. Program jsem úspěšně otestoval s frekvenčním generátorem, ale při testu v motoru nastaly komplikace. Motor sice nastartoval, ale doba otáčky se mezi jednotlivými otáčkami příliš měnila. Výsledkem byl tedy kmitající předstih. Byl jsem tedy nucen promyslet další možnosti snímání. Nejprve mě napadlo přidat druhý impulz ze snímače a počítat tak vše v jedné otáčce. Tuto možnost jsem později zavrhl z důvodu složité implementace. Dalším možným řešením bylo přidat do motoru kódovací disk/inkrementální enkodér a měřit s ním přesně polohu pístu. Tuto možnost jsem úspěšně otestoval se setrvačníkem umístěným na vrtačce, ale již jsem jej nestihl implementovat přímo do motoru. V motoru jsem ale otestoval výkonovou část obvodu pouze se vstupem z hallovy sondy (bez obvodu uC) a s postupnou změnou předstihu od 0° do 90° po 10° jsem změřil výkonové charakteristiky motoru na mnou postavené válcové zkušebně. Pomocí jejich porovnání jsem získal předpokládanou ideální křivku předstihu. Ani tu již jsem z časových důvodů nebyl schopen ověřit. Celé zapalování hodlám dokončit a plně zprovoznit, aby bylo možné jej příští sezónu při závodech použít.

6. SEZNAM OBRÁZKŮ

7. SEZNAM TABULEK

8. SEZNAM PŘÍLOH

Příloha 1: Seznam potřebných součástek

Příloha 2: cinan_novnobeh.xlsx

Příloha 3: krivka_ideal.domzap

Příloha 4: krivka_vypocet.xlsx

Příloha 5: krivky_vykonu.zip

Příloha 6: obrazky.zip

Příloha 7: Programy_PC.zip

Příloha 8: Programy_uC.zip

Příloha 9: Přechodové_char._cívek.zip

Příloha 10: schéma_+_deska.zip

Příloha 11: schémata.zip

Příloha 12: test_krivky_enkoder.xlsx

Příloha 13: test_krivky.domzap

Příloha 14: test_krivky1.xlsx

Příloha 15: testkrivky1.domzap

Příloha 16: uhlomer.SLDPRT

9. LITERATURA

[1] Federal Mogul, BERU: Vše o zapalovacích cívkách. Dostupné na URL: http://beru.federalmogul.com/sites/default/files/ti07_ignition_coils_cz_2014.pdf

[2] Vlastimil Vávrů: Zapalování a seřizování předstihu. Dostupné na URL: http://www.velorexy.cz/clanek/zapalovani-a-serizovani-predstihu

[3] BU931 datasheet. Dostupný na URL: http://www.tme.eu/cz/Document/b3585db555b7397d2b1743d4d5a06c5c/BU931T.pdf

[4] PIC18F2550 datasheet. Dostupný na URL: http://ww1.microchip.com/downloads/en/DeviceDoc/39632b.pdf

[5] Heat Sink Temperature Calculator. Dostupný na URL http://www.daycounter.com/Calculators/Heat-Sink-Temperature-Calculator.phtml

[6] Chladič HS-S01. Dostupný na URL: http://www.tme.eu/cz/details/hs-s01/chladice/stonecold/

[7] Chladič HS-142-50. Dostupný na URL: http://www.tme.eu/cz/details/hs-142-50/chladice/stonecold/

[8] BS170 datasheet. Dostupný na URL: https://www.fairchildsemi.com/datasheets/BS/BS170.pdf

[9] LF50CDT datasheet. Dostupný na URL: http://www.tme.eu/cz/Document/7ef951ff8781134d4e0fc54509f46fc7/lfxx.pdf

[10] Hallova sonda TLE4945L – datasheet. Dostupný na URL: http://www.gme.cz/img/cache/doc/533/095/tle4945l-datasheet-1.pdf

10. PŘÍLOHY

Příloha 1: Seznam součástek

Part	Value	Device	Package
C1	220n	C-EUC0603	C0603
C2	2.2u	CPOL-EUB	PANASONIC_B
C3	0.1u	C-EUC0603	C0603
C4	220n	C-EUC0603	C0603
C5	15p	C-EUC0603	C0603
C6	15p	C-EUC0603	C0603
C7	220n	C-EU150-084X183	C150-084X183
C8	230u/35V	CPOL-EUE2,5-6E	E2,5-6E
C9	100n	C-EUC0402	C0402
D1	1N4933	1N4933	DO41-10
D2	S2K	GF1	SMA-DO214AC
IC1	LF50CDT	L78MXX	DPACK
IC2	PIC18F2550_28W	PIC18F2550_28W	SO28W
J1	PIN1-2	PIN1-2	62221
KK1	SK104-PAD	SK104-PAD	SK104-PAD
LED1	2.4V 10mA	LED3MM	LED3MM
OK1	PC817SMD	PC817SMD	DIL4-SMD
Q1	20M	CSM-7X-DU	CSM-7X-DU
R2	270	R-EU_R0402	R0402
R3	10k	R-EU_R0402	R0402
R4	8.66k	R-EU_R0402	R0402
R5	5k	R-EU_R0402	R0402
R6	300	R-EU_R0402	R0402
R7	150	R-EU_R0402	R0402
R8	82	R-EU_R0402	R0402
R9	8k2	R-EU_R0402	R0402
SV3	PWR	MT6-3	MT6-3
T1	BC337	BC337	TO92
T2	BC337	BC337	TO92
U$3	BU931TV	BU931TV	TO220V
X2	MINI-USB-32005-301	MINI-USB-32005-301	32005-301
X3	snimac	MPT4	4POL254

Zbylé přílohy jsou uloženy zvlášť.