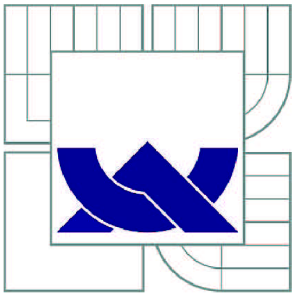
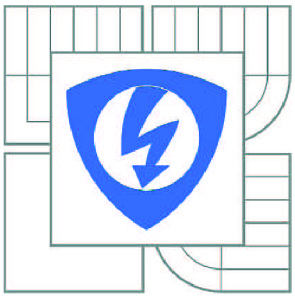


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF RADIO ELECTRONICS

AUTOMATICKÝ ANTÉNNÍ TUNER S INTELIGENTNÍM ALGORITMEM LADĚNÍ

AUTOMATIC ANTENNA TUNER WITH INTELLI-FIT ALGORITHM

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

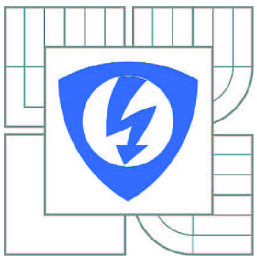
AUTOR PRÁCE — PETR FRECER

AUTHOR

VEDOUCÍ PRÁCE — Ing. ZBYNĚK LUKEŠ, Ph.D.

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav radioelektroniky

Bakalářská práce

bakalářský studijní obor
Elektronika a sdělovací technika

Student: Petr Frecer ***ID:*** 106435
Ročník: 3 ***Akademický rok:*** 2009/2010

NÁZEV TÉMATU:

Automatický anténní tuner s inteligentním algoritmem ladění

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte literaturu týkající se impedančního přizpůsobení antén. Prostudujte příručky profesionálních anténních tunerů (MFJ, Elecraft, atd.). Navrhněte koncepci anténního tuneru pro zadané pásmo. Navrhněte malý přenosný tuner řízený mikroprocesorem pro výkony do 50 W. Realizujte funkční vzorek, jeho vlastnosti ověřte.

Napište program do ovládací jednotky procesoru ATMEGA tuneru v jazyku C. Vytvořte inteligentní algoritmus, který bude rychle a efektivně přizpůsobovat impedanci vysílače k impedanci antény.

DOPORUČENÁ LITERATURA:

[1] PROCHÁZKA, M. Antény - Encyklopedická příručka. Praha: BEN - technická literatura, 2000.

[2] IntelliTuner: Automatic Antenna Tuner [online]. MFJ Enterprises [cit. 15. 5. 2009]. Dostupné na: http://www.mfjenterprises.com/man/pdf/MFJ-993.pdf

[3] Elecraft: Hands-On Ham Radio [online]. Elecraft [cit. 15. 5. 2009]. Dostupné na www: http://www.elecraft.com/

Termín zadání: 8.2.2010 ***Termín odevzdání:*** 28.5.2010

Vedoucí práce: Ing. Zbyněk Lukeš, Ph.D.

prof. Dr. Ing. Zbyněk Raida
Předseda oborové rady

ABSTRAKT

Bakalářská práce se zabývá problematikou impedančního přizpůsobení antén, návrhem a realizací automatického anténního tuneru. Práce uvádí základní fyzikální vztahy pro popis odrazů na elektrickém vedení a impedančního přizpůsobení. Dále jsou zvažována různá obvodová řešení LC článků jako transformátorů impedance. Je vybráno řešení vyhovující zadaným požadavkům. Konstruované zařízení je navrženo s ohledem na plně automatickou činnost. Všechny činnosti jsou řízeny pokyny mikrokontroléru. Rychlé přizpůsobení impedance zabezpečuje inteligentní algoritmus. Jako nezbytný zdroj vstupních dat slouží PSV-metr. Mikrokontrolér je dále doplněn o periferie rozšiřující funkčnost zařízení (frekvenční čítač, externí paměť). Ovládání tuneru je realizováno dálkově pomocí programu v osobním počítači komunikujícím přes sběrnici USB nebo RS-232.

KLÍČOVÁ SLOVA

Impedanční přizpůsobení antény, automatický anténní tuner, PSV-metr, frekvenční čítač, mikrokontrolér ATmega16.

ABSTRACT

The bachelor's thesis deals with an antenna impedance matching, a design and a construction of an automatic antenna tuner. Basic physical equations are mentioned describing reflections on a feed line and the impedance matching. Various LC networks are discussed for their impedance-transforming possibility. One of the designs is chosen according to given requirements. A system is designed as fully-automated. All activities are managed by a microcontroller. Fast impedance matching is provided by an intelligent algorithm. A SWR-meter serves as a vital data source. The system is equipped with additional circuits that improve functionality (a frequency counter, an external memory). The system can be controlled remotely by a personal computer programme communicating via USB or RS-232.

KEYWORDS

Antenna impedance matching, automatic antenna tuner, SWR-meter, frequency counter, ATmega16 microcontroller.

FRECER, P. *Automatický anténní tuner s inteligentním algoritmem ladění*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Ústav radioelektroniky, 2010. 39 s., 26 s. příloh. Bakalářská práce. Vedoucí práce: Ing. Zbyněk Lukeš, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma Automatický anténní tuner s inteligentním algoritmem ladění jsem vypracoval samostatně pod vedením vedoucího 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této 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 § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

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

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Bakalářská práce se zabývá návrhem a realizací automatického anténního tuneru pro použití na krátkých vlnách (kmitočet 3 – 30 MHz) a pro výkony do 50 W. Anténní tuner je obvod realizující impedanční přizpůsobení antény k napáječi. Funguje jako transformátor impedance.

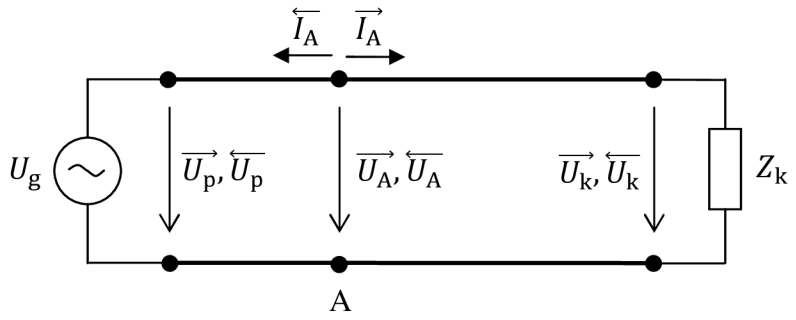
Úvod do teorie vedení a impedančního přizpůsobení je zpracován v první kapitole. Následuje rozbor vhodnosti různých zapojení elektrických filtrů transformujících impedanci pro dané zadání a jejich uplatnění v praxi. Třetí kapitola popisuje konkrétní obvodový návrh a realizaci anténního tuneru. Čtvrtá kapitola popisuje software řídící jednotky anténního tuneru včetně inteligentního algoritmu ladění. V páté kapitole je popsáno uživatelské rozhraní aplikace pro osobní počítač dálkově ovládající tuner. Předposlední kapitola se zabývá ověřením funkčnosti tuneru a srovnává tunerem měřené hodnoty s údaji poskytovanými profesionálními měřicími systémy. Poslední kapitola shrnuje dosažené výsledky a diskutuje splnění zadání bakalářské práce.

1 KLASICKÁ TEORIE VEDENÍ

Pro dobré porozumění následujícím kapitolám a samotné funkci anténního tuneru shrnuje tato kapitola základní pojmy a definice z klasické teorie vedení (viz. [1]), která se zabývá popisem rozložení napětí a proudu na vedení. V následujícím textu se uvažuje použití dvouvodičového vedení.

1.1 Odrazy na vedení

Dvouvodičové vedení napájené zdrojem harmonického napětí U_g je zakončeno impedancí zátěže Z_k (obr. 1.1). Směrem od zdroje k zátěži se šíří tzv. *přímá (postupná) vlna* $\overrightarrow{U}, \overrightarrow{I}$. V případě obecné impedance zátěže se část energie na konci vedení odrazí a opačným směrem se šíří *zpětná (odražená) vlna* $\overleftarrow{U}, \overleftarrow{I}$. Napětí i proud přímé i odražené vlny se při obecné zátěži mění podél délky vedení periodicky s periodou rovnou polovině vlnové délky budícího signálu.



Obr. 1.1: Vedení zakončené impedancí Z_k

Pro konkrétní bod na vedení je poměr napětí postupné vlny v tomto bodě a proud postupné vlny tekoucí tímto bodem roven tzv. *charakteristické impedanci* Z_{ov} dané vztahem

$$\frac{\overrightarrow{U_A}}{\overrightarrow{I_A}} = \frac{\overleftarrow{U_A}}{\overleftarrow{I_A}} = Z_{ov}. \tag{1.1}$$

Stejná závislost platí i pro poměr napětí a proudu odražené vlny. Charakteristická impedance je konstantní podél celého vedení a je závislá na elektrických vlastnostech samotného vedení, čili jakým způsobem a z jakých materiálů je vedení vyrobeno. Dvouvodičová vedení se obvykle konstruují s charakteristickou impedancí 50 Ω.

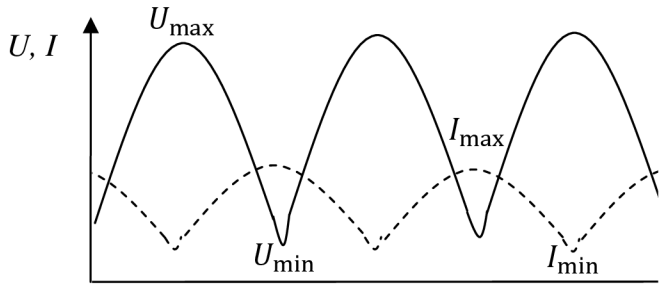
Poměr odražené a přímé vlny napětí na konci vedení určuje *činitel odrazu* ρ_k na konci vedení:

$$\rho_k = \frac{\overleftarrow{U_k}}{\overrightarrow{U_k}} = \frac{Z_k - Z_{ov}}{Z_k + Z_{ov}}. \tag{1.2}$$

V rovnici (1.2) je uvedena souvislost činitele odrazu s charakteristickou impedancí a impedancí zátěže. V případě rovnosti $Z_k = Z_{ov}$ je činitel odrazu nulový. Na vedení nevznikne odražená vlna a veškerá energie přenášená vedením se přenese na zátěž. Takové vedení se nazývá *impedančně přizpůsobené*.

1.2 Stojatá vlna na vedení

V předchozím popisu je začátek vedení s budícím zdrojem považován za impedančně přizpůsobený. Pokud tomu tak není, dojde k odrazu části zpětné vlny na začátku vedení a vzniklá vlna se sečte s přímou vlnou a společně se šíří k zátěži, kde dojde opět k odrazu. Na vedení takto postupně vznikne velké množství vln šířících se k zátěži nebo od ní. Jejich složením vznikne tzv. *stojatá vlna*, jejíž rozložení napětí a proudů se mění podél délky vedení (obr. 1.2), nemění se ale v čase.



Obr. 1.2: Rozložení napětí a proudu stojaté vlny podél vedení

V místech, kde je přímá vlna ve fázi se zpětnou vlnou dochází ke vzniku maxim – *kmiten*, naopak v místech, kde jsou obě vlny v protifázi, vznikají minima – *uzly*:

$$U_{max} = \left|\vec{U}\right| + \left|\overleftarrow{U}\right|, \tag{1.3}$$

$$U_{min} = \left|\vec{U}\right| - \left|\overleftarrow{U}\right|. \tag{1.4}$$

Poměr napětí v kmitně U_{max} a v uzlu U_{min} určuje *poměr stojatých vln* (PSV, anglicky označován jako SWR – Standing Wave Ratio) daný vztahem

$$\sigma = \frac{U_{max}}{U_{min}} = \frac{\left|\vec{U}\right|+\left|\overleftarrow{U}\right|}{\left|\vec{U}\right|-\left|\overleftarrow{U}\right|} = \frac{1+|\rho|}{1-|\rho|}. \tag{1.5}$$

V případě, že je činitel odrazu maximální, tj. rovný jedné a na konci vedení se veškerá energie odrazí, nabývá poměr stojatých vln nekonečnou hodnotu. V opačném extrému, kdy je vedení přizpůsobené, je činitel odrazu nulový a poměr stojatých vln rovný jedné ($1 \leq \sigma < \infty$).

1.3 Impedanční přizpůsobení antény

Anténa, ať už vysílací nebo přijímací, je k navazujícím elektrickým obvodům připojena pomocí tzv. *napáječe*. Toto vedení můžeme popsat výše zmíněnou terminologií. Má vlastní charakteristickou impedanci a anténa je k němu připojena jako zátěž. Impedance antény je obecně komplexní a mění se v závislosti na pracovním kmitočtu. Z hlediska efektivního přenosu energie při vysílání / příjmu je třeba, aby byla anténa k vedení impedančně přizpůsobená.

Antény jsou konstrukčně řešeny tak, aby na určitém pracovním kmitočtu rezonovaly. Jejich impedance je pak čistě reálná se standardní hodnotou (např. 50 Ω). Napáječ má většinou shodnou charakteristickou impedanci stejně jako navazující elektrické obvody. Není proto potřeba anténu impedančně přizpůsobovat. Při použití různých pracovních kmitočtů je nutné používat více antén navržených na konkrétní kmitočty. Pokud je dostupná jen jedna anténa, objevuje se problém jejího impedančního přizpůsobení na kmitočtech mimo její rezonanci.

Impedanční přizpůsobení antény je možné provést pomocí úseku přizpůsobovacího vedení, zavedením přídavné diskontinuity nebo pomocí impedančního transformátoru (viz. [2]). Poslední řešení je možné realizovat širokopásmově pomocí proměnných prvků indukčnosti nebo kapacity. Anténa tak může pracovat s různými pracovními kmitočty bez potřeby návrhu a konstrukce přizpůsobení pro každý pracovní kmitočet zvlášť.

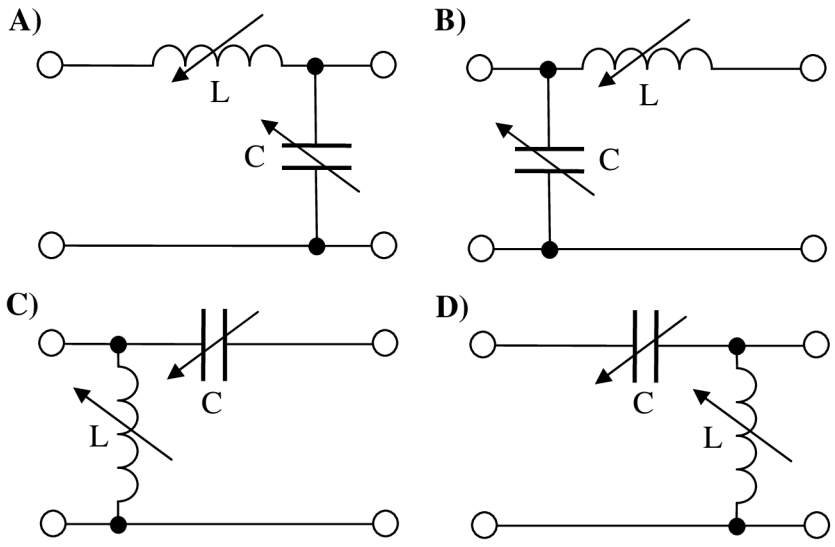
Jako kriterium přizpůsobení je obvykle brána hodnota poměru stojatých vln, která by měla být pro vysílací antény $\sigma \leq 2$, v případě přenosu elektromagnetické vlny se širokopásmovou modulací nebo při velmi dlouhém napáječi $\sigma \leq 1{,}1$. Samotný přizpůsobovací obvod by měl být umístěn co nejblíže anténě, protože vlivem útlumu vedení se velikost postupné vlny směrem k anténě snižuje a stejně tak se utlumuje odražená vlna šířící se opačným směrem. Hodnota poměru stojatých vln na začátku vedení je tak odlišná od hodnoty na konci (u antény). Vedení se na začátku jeví jako lépe přizpůsobené, přestože není.

2 IMPEDANČNÍ TRANSFORMÁTORY

Za impedanční transformátory se považují obvody se schopností transformovat impedanci připojenou k výstupu na požadovanou hodnotu impedance na vstupu. Možnosti konstrukce takového transformátoru jsou velké. Následující text se omezuje na nejjednodušší příčkové články LC, které jsou převzaty z oblasti návrhu pasivních elektrických filtrů (viz. [3]). Elektrické filtry se kromě svého primárního účelu – filtrace elektrického signálu vyznačují také transformací zatěžovací impedance. Mezi popisované struktury byl vybrán článek L, článek T a článek Π, neboť tyto články jsou nejčastěji používány v komerčně vyráběných anténních tunerech.

2.1 L článek

Nejjednodušší zapojení představuje L článek. Skládá se jen ze dvou proměnných prvků L a C. Na obr. 2.1 jsou zobrazeny všechny možnosti zapojení. Zapojení A a B představují filtr typu dolní propust, C a D pak horní propust. Použití dolní propusti je výhodné, pokud potřebujeme potlačit vyšší harmonické pracovního kmitočtu, které mohou vznikat ve vysílači. Pro větší širokopásmovost přizpůsobení je vhodné zkombinovat zapojení A a B tak, že se přepínačem připíná kapacita před nebo za indukčnost.



Obr. 2.1: Varianty zapojení L článku

Pro objasnění funkce impedančního přizpůsobení následuje vzorový případ, pro který bylo použito zapojení A z obr. 2.1. Na pracovním kmitočtu f je potřeba přizpůsobit impedanci zátěže $Z_k = R_\mathrm{k} + jX_k$ k charakteristické impedanci vedení $Z_\mathrm{P} = R_\mathrm{p} + jX_\mathrm{p}$. Hodnoty reaktancí prvků L a C udávají následující rovnice (převzato z [4]).

$$X_{\mathrm{L}} = \sqrt{\left(R_{\mathrm{p}} + jX_{\mathrm{p}}\right)\left[\left(R_{\mathrm{p}} + jX_{\mathrm{p}}\right) - \left(R_{\mathrm{k}} + jX_{k}\right)\right]}, \tag{2.1}$$

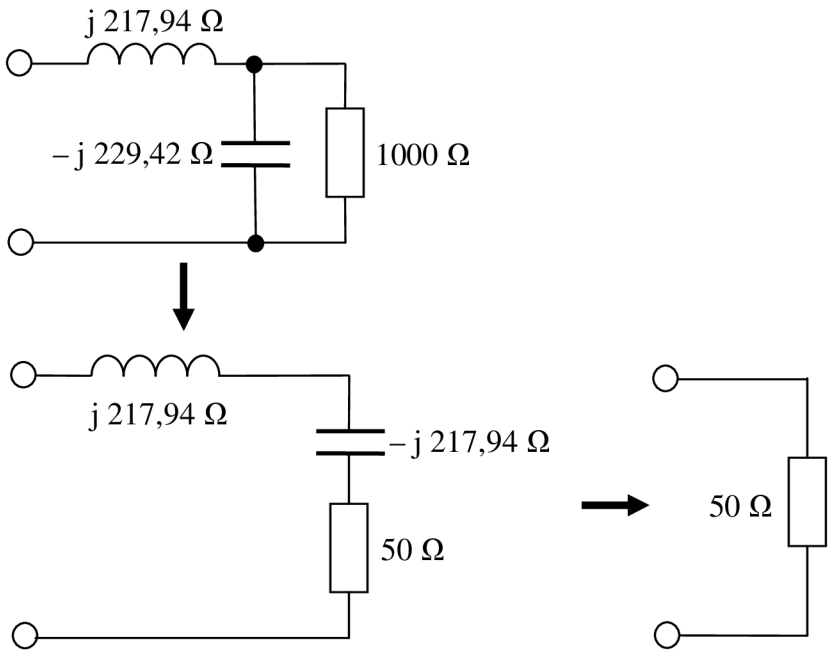
$$X_{\mathrm{C}} = \left(R_{\mathrm{k}} + jX_{\mathrm{k}}\right)\sqrt{\frac{R_{\mathrm{p}} + jX_{\mathrm{p}}}{\left(R_{\mathrm{k}} + jX_{\mathrm{k}}\right) - \left(R_{\mathrm{p}} + jX_{p}\right)}} \tag{2.2}$$

Z reaktancí se následovně určí hodnoty indukčnosti L a kapacity C.

$$C = \frac{1}{2\pi f X_{\mathrm{C}}}, \tag{2.3}$$

$$L = \frac{X_{\mathrm{L}}}{2\pi f}. \tag{2.4}$$

Pro konkrétní zadání f = 28 MHz, Zp = 50 Ω a Zk = 1000 Ω jsou obdržené výsledky X_{L} = j 217,94 Ω, X_{C} = – j 229,42 Ω, C = 24,78 pF, L = 1,239 μH. Postupným zjednodušováním obvodu znázorněným na obr. 2.2 jsou L článek a zatěžovací impedance nahrazeny výslednou impedancí, která zakončuje vedení.

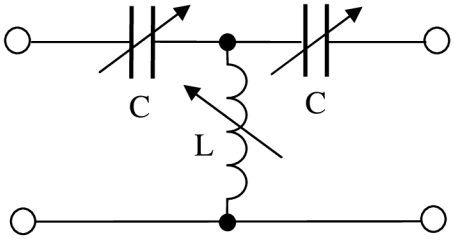


Obr. 2.2: Postup určování náhradní impedance

Prvním krokem zjednodušování (viz. [5]) byl přepočet impedancí paralelních prvků na admitance a jejich sečtení. Následně se výsledná admitance převedla zpět na impedanci. Posledním krokem byl součet impedancí prvků v sérii. Výsledná impedance odpovídá zadání. Anténa je přizpůsobena k vedení.

2.2 T článek

Doplněním L článku o další laditelný prvek vznikne T článek (obr. 2.3). Přizpůsobovací obvod se díky tomuto rozšíření stává více univerzální. Dokáže impedančně přizpůsobit větší rozsah impedancí. Na rozdíl od L článku má ale více možných kombinací hodnot prvků, které přizpůsobí impedanci antény na požadovanou hodnotu.

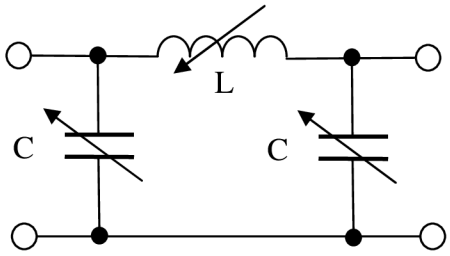


Obr. 2.3: T článek

Nevýhodou T článku jsou vyšší ztráty přenášeného výkonu. Ztráty jsou ovlivněny zvolenou kombinací prvků. Pro rozdílné kombinace se mohou lišit až o desítky procent. Přesto se T článek objevuje ve většině komerčně vyráběných anténních tunerů.

2.3 Π článek

Π článek se podobá předcházejícímu T článku, sdílí stejné výhody i nevýhody. Navíc se chová jako filtr typu dolní propust a utlumuje tak vyšší harmonické pracovního kmitočtu.



Obr. 2.4: Π článek

3 NÁVRH A REALIZACE ANTÉNNÍHO TUNERU

3.1 Volba koncepce tuneru

Existují dvě základní koncepce při konstruování anténního tuneru. První je tuner ovládaný manuálně (obr. 3.1). Pro změnu hodnot kapacity využívá otočné kondenzátory a indukčnost se mění přepínáním odboček cívky. Obsluha hodnotí dosažené impedanční přizpůsobení pomocí PSV-metru (měřiče poměru stojatých vln). PSV-metr může být externí nebo přímo vestavěn v anténním tuneru.

Obr. 3.1: Manuální anténní tuner obsahující T článek a vestavěný PSV-metr

Druhou koncepcí je tuner automatický. Na základě dat poskytnutých PSV-metrem rozhoduje řídící jednotka (mikrokontrolér) jakou má vybrat hodnotu indukčnosti a kapacity. Hodnoty lze vybírat obdobně jako u manuálního tuneru doplněním otočných prvků o servomotory nebo připínáním požadovaných hodnot pomocí relé.

Obr. 3.2: Automatický anténní tuner Elecraft KAT100 bez horního krytu (převzato z [6]).

Ze zadání této práce plyne požadavek na konstrukci tuneru automatického.

Za tímto účelem byly prostudovány příručky několika profesionálních anténních tunerů amerických výrobců Elecraft a MFJ Enterprises (viz. [6], [7], [8], [9]).

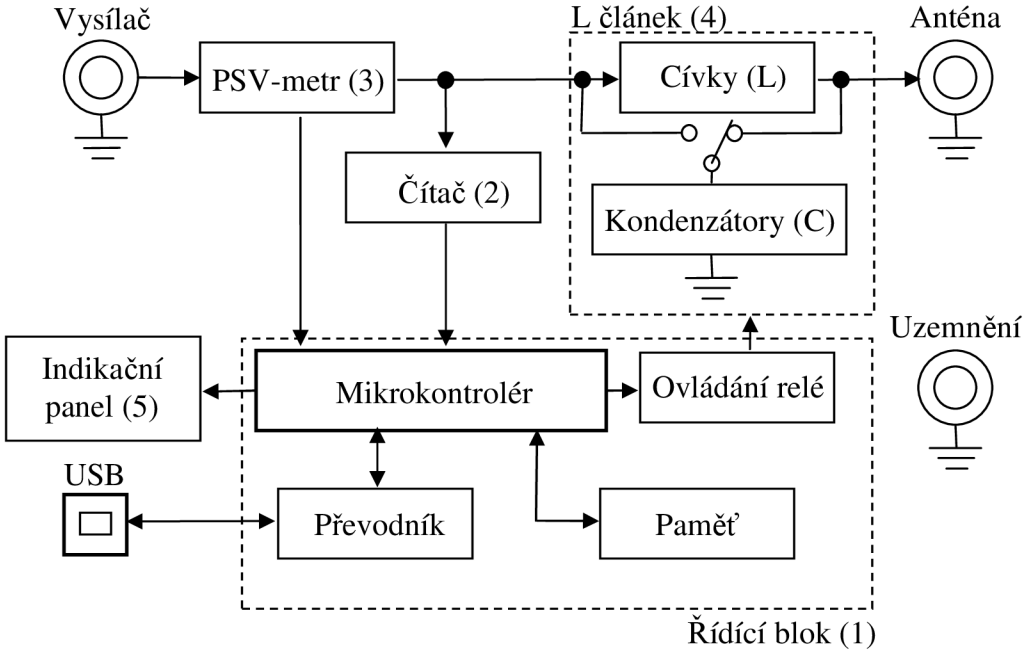
Společnými rysy těchto tunerů je použití L článku s možností připojení kapacity před nebo za indukčnost. Indukčnost je realizována souborem toroidů s rostoucím počtem závitů. Kapacita je dána řadou kondenzátorů. Potřebné hodnoty vybírají relé. Příkladem může být automatický anténní tuner Elecraft KAT100 na obr. 3.2.

L článek je pro automatizaci vhodnější z hlediska menší složitosti zapojení. Výsledkem přizpůsobení je jen jedna možná kombinace prvků L a C, což zjednodušuje řídící proces. Nevýhoda L článku v manuálních tunerech – omezený rozsah otočných kondenzátorů – se při použití více připínaných kondenzátorů neprojeví. Z těchto důvodů jsou na základě uvedené koncepce založeny následující kroky návrhu automatického tuneru.

Konkrétní návrh dále podléhá požadavkům zadání. Tuner má přizpůsobovat impedanci antény v oblasti kmitočtů krátkých vln. Má být lehce přenositelný a dimenzován pro výkony do 50 W. Z úvahy v kapitole 1.3 vyplynul požadavek na umístění tuneru v blízkosti antény. Anténa je většinou umístěna mimo dosah obsluhy. Proto bude navrhovaný tuner ovládán dálkově.

3.2 Funkční bloky tuneru

Funkci navrhovaného obvodu popisuje blokové schéma na obr. 3.3. Obsahuje řídící blok (1), čítač (2), PSV-metr (3), L článek (4), indikační panel (5) a napájecí obvody (6). Mikrokontrolér rozhoduje na základě informací z PSV-metru, jaké má použít hodnoty L a C. Výběr hodnot je proveden prostřednictvím obvodů spínajících příslušná relé.



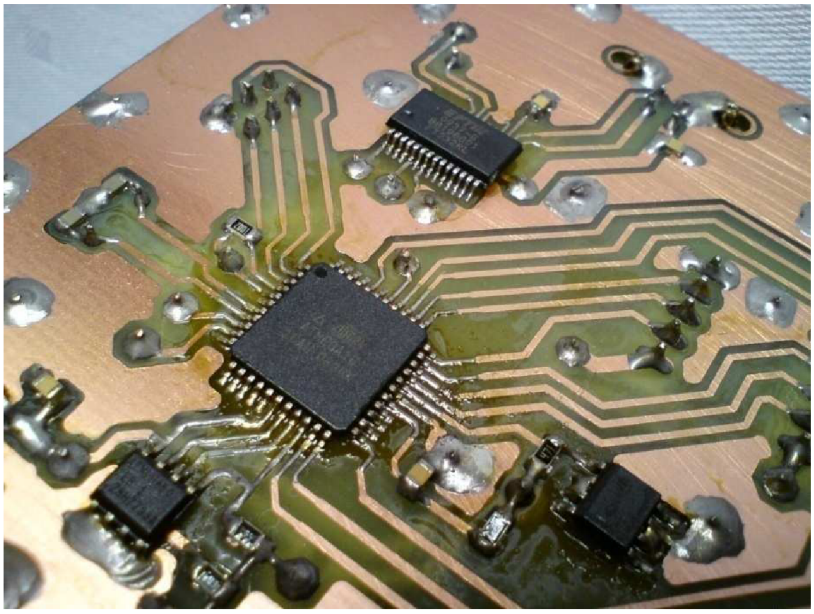
Obr. 3.3: Blokové schéma navrhovaného anténního tuneru

Po dokončení impedančního přizpůsobení je vyhledaná kombinace L a C pro budoucí použití uložena do paměti společně s údajem pracovního kmitočtu. Kmitočet mikrokontrolér zjišťuje prostřednictvím čítače. Případné pokyny kontrolér přijímá přes sběrnici USB (Universal Serial Bus) nebo RS-232. Pracovní stav tuneru je indikován na panelu z LED diod.

3.3 Řídící blok

Základ řídícího bloku (viz. obr. 3.4) tvoří mikrokontrolér. Kritérii při jeho výběru byl dostatečný počet vstupně/výstupních pinů, přítomnost externě řízeného čítače, minimálně dvoukanálový A/D převodník, hodinový kmitočet vyšší jak 15 MHz, možnost sériové komunikace a dostatečně velká programová paměť. Vybrán byl osmibitový mikrokontrolér ATmega16 fy Atmel Corporation (viz. [10]) s 16 kB programové paměti.

Obvodové zapojení řídícího bloku lze nalézt v příloze A.1. Napájecí napětí bylo zvoleno 5 V. Kmitočet hodinového signálu určuje externí oscilátor s krystalem 16 MHz. Programování je umožněno přes sériové rozhraní SPI (Serial Peripheral Interface). Signály z příslušných pinů SPI (na portu B) jsou vyvedeny na šestipinový programovací konektor SV1.



Obr. 3.4: Realizace řídící části na desce plošných spojů

Na dolní dva piny portu A disponující A/D převodníky jsou přivedena analogová data poskytována PSV-metrem. Pro stabilnější výsledky A/D převodu je převáděná hodnota porovnávána s napěťovou referencí 2,5 V připojenou na pinu AREF, kterou zprostředkovává integrovaný obvod TL431. Na zbylých šesti pinech portu A je připojeno ovládání indikačního panelu LED a signály pro možné budoucí rozšíření funkční nabídky tuneru.

Na prvním pinu portu B je připojen vstup hodinového signálu, který poskytuje

obvod čítače. Obdélníkový signál řídí vnitřní čítač mikrokontroléru a umožňuje měřit pracovní kmitočet, na kterém probíhá impedanční přizpůsobení antény. Port C zprostředkovává komunikaci mikrokontroléru s externí pamětí a s obvody spínajícími relé.

Požadavek na použití externí paměti je vyvolán potřebou uchování dat o přizpůsobení antény. Při příštím ladění antény na dříve použitém pracovním kmitočtu dojde k načtení příslušných hodnot L a C a tím se výrazně urychlí doba přizpůsobování impedance. Data musí být v paměti uchována i při odpojeném napájení, což umožňuje paměť typu EEPROM. Obsažena je i v samotném mikrokontroléru, ale její kapacita je nedostačující. Proto byla vybrána externí paměť s kapacitou 32 kB. Použit byl obvod AT24C256B fy Atmel Corporation, který s mikrokontrolérem komunikuje sériově pomocí komunikace I^2C (Inter Integrated Circuit).

Použitý mikrokontrolér neobsahuje řadič USB sběrnice. Proto je v návrhu použit převodník komunikace z USB na sériovou komunikaci UART (Universal Asynchronous Receiver Transmitter). Komunikaci přes UART umožňuje první dvojice pinů portu D mikrokontroléru. Pro funkci převodníku byl vybrán obvod FT232RL fy FTDI. Je nenáročný z hlediska potřeby připojení externích součástek. Obvodové zapojení (viz. příloha A.1) bylo převzato z údajů od výrobce (viz. [12]). Dva programovatelné vstupně/výstupní piny byly využity pro připojení LED diod, které signalizují příchozí a odchozí komunikaci po sběrnici USB.

3.4 Čítač

Obvod čítače (převzat z [11]) upravuje vstupní vysokofrekvenční signál odebíraný z přizpůsobovaného vedení. Výstupem je obdélníkový signál, který řídí čítač v mikrokontroléru. Obvodové zapojení (viz. příloha A.2) lze rozdělit do tří funkčních bloků. V prvním části dojde k napěťovému omezení vysokofrekvenčního signálu shora i zdola pomocí diod D201, D202 na hodnotu ± 0,6 V. V druhém bloku je signál převeden na obdélníkový pomocí tranzistorového spínače s unipolárním tranzistorem Q2 a zesílen zesilovačem s bipolárním tranzistorem Q3 tak, aby obdélníkový signál nabýval hodnot 0 V až necelých 5 V. Tyto napěťové úrovně odpovídají logické nule a jedničce.

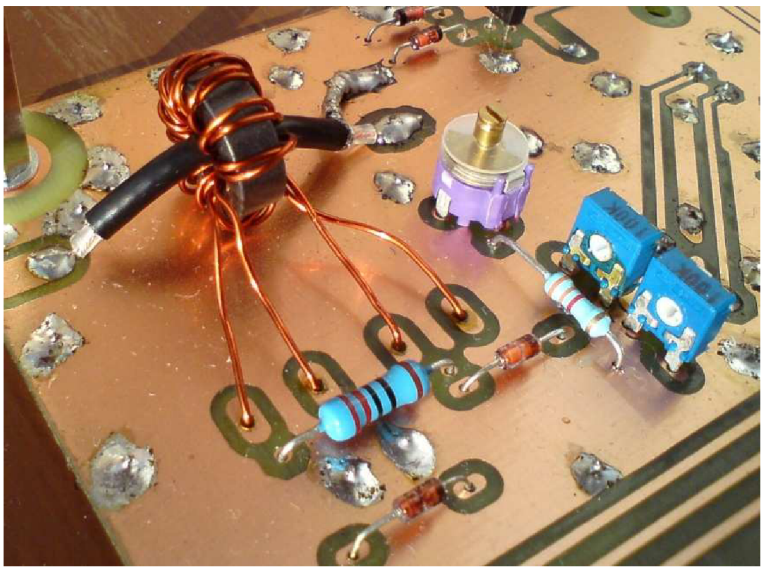
Obr. 3.5: Realizace čítače na desce plošných spojů

Ve třetím bloku je použit dvakrát klopný obvod typu D v zapojení děličky kmitočtu dvěma. Výsledný obdélníkový signál má tak čtyřikrát nižší kmitočet. Snížení hodnoty

kmitočtu je nutné, protože čítač v mikrokontroléru může být řízen jen kmitočtem více jak dvakrát nižším než je hodinový signál mikrokontroléru (16 MHz) a pásmo používaných pracovních kmitočtů antény tuto hodnotu překračuje.

3.5 PSV-metr

Návrh měřiče poměru stojatých vln byl převzat z konstrukce tuneru Elecraft KAT100 (viz. [6]). Obvodové schéma (viz. příloha A.2) představuje upravený tzv. Brueneův můstek. Jeho podstatou je odebrání proudového vzorku z vysokofrekvenčního vedení pomocí transfomátoru SWR1 a napěťového vzorku pomocí kapacitního děliče C305, C306. Oba vzorky se vhodně fázově sečtou a jsou usměrněny diodovým usměrňovačem s rychlou Schottkyho diodou. Transformátor SWR1 se skládá z feritového toroidu (Amidon - materiál 43), primárního vedení tvořeného jedním závitem měřeného vedení a dvou bifilárně navinutých sekundárních vedení. V jedné větvi je výsledné usměrněné napětí úměrné velikosti napětí postupné vlny na vysokofrekvenčním vedení. V druhé větvi je napětí úměrné velikosti odražené vlny. Velikost výstupních napěťových úrovní usměrňovačů je možné upravit odporovými trimry R303, R304. Obě napětí jsou přivedena na A/D převodník mikrokontroléru. Mikrokontrolér výpočtem (viz. rovnice (1.5)) zjistí hodnotu poměru stojatých vln.



Obr. 3.6: Realizace PSV-metru na desce plošných spojů

Pro správnou funkci PSV-metru je nutná jeho kalibrace. Při té se namísto antény zapojí reálná zátěž 50 Ω a kapacitním trimrem C305 se nastaví co nejmenší detekovaná hodnota odražené vlny. Pro tento účel byla vyrobena 50 Ω umělá zátěž (viz. obr. 3.7). Skládá se z konektoru N (zásuvka) a dvou paralelně spojených rezistorů 100 Ω. Celková výkonová zatížitelnost zátěže je 4 W. Měřením na vektorovém obvodovém analyzátoru byla stanovena použitelnost zátěže až do kmitočtu přibližně 200 MHz. Pro kalibraci v pásmu krátkých vln proto vyhovuje.

Obr. 3.7: 50 Ω umělá zátěž (P_{max} = 4 W)

3.6 L článek

Použití relé pro spínání indukčnosti a kapacity přináší kromě dříve zmiňovaných výhod i malou nevýhodu. Je jí nespojitá změna těchto parametrů. Mohou se měnit jen s určitým krokem daným nejmenší realizovanou indukčností a kapacitou. Pro pokrytí velkého rozsahu hodnot a zachování malého kroku byly hodnoty parametrů L a C voleny tak, že každá následující hodnota je dvojnásobkem předchozí. Hodnot každého parametru je celkem sedm, z čehož vyplývá 128 kombinací každého parametru a 16384 možných kombinací L a C.

Indukčnost nabývá hodnot od 0,0 μH do 10,3 μH. Kapacita je v rozsahu 0 pF až 1270 pF. Zvolené hodnoty udávají tabulky tab. 3.1 a tab. 3.2. Obvodové zapojení je v příloze A.3. První dvě hodnoty indukčnosti jsou realizovány vzdušnými cívkami. Zbývajících pět hodnot vytváří pět vinutí na jednom toroidu (obr. 3.8). Jádro toroidu bylo vybráno, s ohledem na rozsah použitých kmitočtů, železoprachové od fy Amidon. Konkrétní typ T157-2 byl vybrán podle rozměrů s pomocí vzorce pro potřebný počet závitů cívky, který udává výrobce:

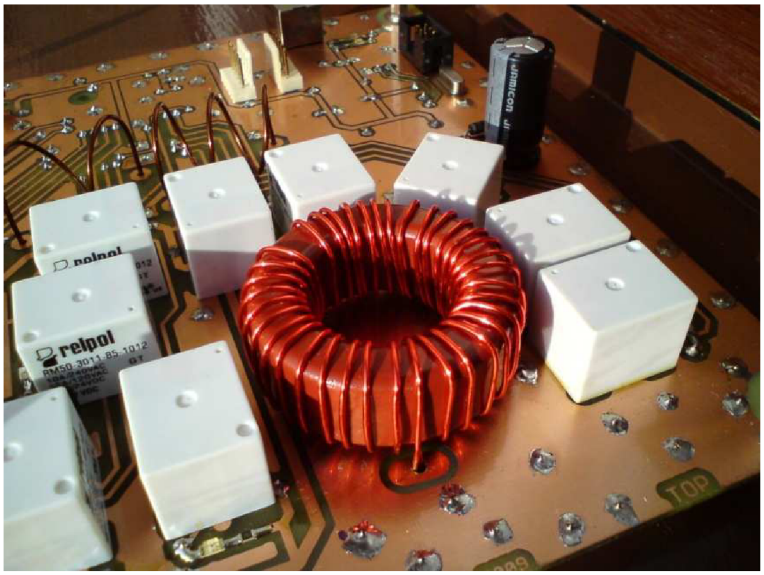
$$N = 100\sqrt{\frac{L[\mu\mathrm{H}]}{A_\mathrm{L}\left[\frac{\mu\mathrm{H}}{100\ \mathrm{z.}}\right]}}. \tag{3.1}$$

Měření indukčnosti pomocí Q-metru ukázalo, že vzorec (3.1) neuvažuje vliv dalších vinutí na toroidu na výsledný počet závitů. Počet závitů jednotlivých vinutí byl podle měřených výsledků upraven (viz. tab. 3.1).

Tab. 3.1: Použité hodnoty indukčnosti a odpovídající počet závitů cívek

Indukčnost [µH]	Vypočtený počet závitů	Použitý počet závitů
0,08	-	2
0,16	-	3
0,32	4,78	3
0,64	6,76	4
1,3	9,64	6
2,6	13,63	9
5,2	19,27	14

Příklad výpočtu dle vzorce (3.1): $N = 100\sqrt{\frac{0{,}08}{140}} = 4{,}78$ z.



Obr. 3.8: Cívky na toroidním jádře

Pro realizaci hodnot kapacity byly vybrány slídové kondenzátory. Mají malou výrobní odchylku od nominální hodnoty (5%) a snesou zatížení napětím ve stovkách voltů. Sestavení požadovaných hodnot z komerčně dostupných součástek udává tab. 3.2.

V zapojení jsou použita relé RM50 dimenzovaná na spínaná napětí do 240 V. Řízena jsou stejnosměrným napětím 12 V. Obvodové zapojení řízení relé popisuje příloha A.4. Paralelně k cívce každého relé je připojena žlutá LED dioda indikující sepnutý stav. Antiparalelně je zapojena dioda chránící obvody před vysokým napětím, které vzniká na cívce relé při rozepnutí.

Tab. 3.2: Použité hodnoty kapacity

Požadovaná kapacita [pF]	Kombinace dostupných hodnot [pF]	Použitá kapacita [pF]
10	10	10
20	22	22
40	33 + 10	43
80	68 + 10	78
160	150	150
320	250 + 68	318
640	500 + 150	650

Spínání relé obstarávají posuvné registry IC5, IC6. Rozšiřují počet výstupních pinů mikrokontroléru a umožňují jejich větší proudové zatížení. Použity jsou posuvné registry TPIC6C596PW obsahující osm spínačů s výkonovými tranzistory DMOS.

3.7 LED panel

Provozní stavy anténního tuneru indikuje LED panel. Je realizován na samostatné desce plošných spojů. Obvodové zapojení (viz. příloha A.6) se skládá z konektoru pro spojení pětižilovým kabelem s deskou tuneru a tří LED diod. Ke každé diodě je v sérii zapojen rezistor omezující proud diodou na přibližně 5 mA. Zelená dioda POWER indikuje stav napájení tuneru (zapnuto/vypnuto). Žlutá dioda BUSY oznamuje, že tuner právě vykonává nějakou operaci. A červená dioda TUNE indikuje probíhající proces automatického ladění.

3.8 Napájecí obvody

Hodnotu napájecího napětí tuneru určují relé, které vyžadují 12 V stejnosměrné napětí. Zapojení napájecích obvodů ukazuje příloha A.5. Mikrokontrolér a další logické obvody pracují s napájecím napětím 5 V. Vytvořeno je stabilizátorem LM78M05CDT schopným dodat proud až 0,5 A, který je zapojený dle doporučení výrobce (viz. [13]). Před následky nechtěného přepólování zdroje napětí chrání logické obvody dioda D601. Filtraci napájecího napětí provádí elektrolytický kondenzátor C601 a keramický kondenzátor C602 přímo u vstupu 12 V napětí na desku plošných spojů. Každý integrovaný obvod je navíc doplněn o keramický kondenzátor s hodnotou 100 nF.

3.9 Konstrukce tuneru

Na základě návrhu obvodového zapojení (přílohy A.1 až A.6) byly zvoleny odpovídající elektronické součástky (příloha A.15) a byly pro ně vypracovány dvě dvouvrstvé desky plošných spojů. Vyhotovení hlavní desky plošných spojů a její osazení součástkami dokumentují přílohy A.7, A.9, A.11 a A.13. Pro minimalizaci šíření rušivých signálů

mezi jednotlivými spoji je deska z obou stran vybavena tzv. rozlitou zemní vrstvou. Obě zemní vrstvy jsou spojeny četnými prokovy. Druhou desku plošných spojů (indikační panel) popisují přílohy A.8, A.10, A.12 a A.14.

Dle rozměrů hlavní desky (170 x 180 mm) byla pro anténní tuner zkonstruována a vyrobena krabička. Krabička o vnějších rozměrech 171 x 181 x 37 mm je oporou pro desky plošných spojů. Zabraňuje kontaktu s nežádoucími okolními předměty a částečně stíní vnitřní prostor tuneru před rušivými signály z okolí. Krabička je vyrobena z pocínovaného ohýbaného plechu. Jednotlivé díly jsou spojeny pájkou kromě horního víka, které je přišroubované. K hornímu víku je přichycen indikační panel. Víko je kryto popisným štítkem nesoucím informace o jednotlivých ovládacích prvcích a konektorech. Tuner je vybaven dvěma vysokofrekvenčními konektory N pro vstupní a výstupní signál, konektorem pro stejnosměrné napájení, konektorem USB a přepínačem pro zapnutí napájení. Výsledný vzhled anténního tuneru je na obr. 3.9.

Obr. 3.9: Vnější pohled na sestavený anténní tuner

4 SOFTWARE ŘÍDÍCÍ JEDNOTKY

Ovládací program mikrokontroléru ATmega16 spravuje veškerou činnost tuneru. Kvůli rozsahu vytvářené aplikace bylo zvoleno programování ve vyšším programovacím jazyce. Byl vybrán programovací jazyk C (viz. [14]). K vytváření programu bylo využito vývojové prostředí AVR Studio 4.18 (viz. [15]) a balík programů WinAVR (viz. [16]) s knihovnami funkcí avr-libc 1.67 (viz. [17]). Nahrání programu do programové paměti mikrokontroléru obstaral programátor BiProg, který byl vyroben dle návodu [18]. BiProg používá programování pomocí ISP (In-System Programming) a je možné ho spojit s deskou tuneru pomocí šestipinového konektoru SV1 (viz. schéma A.1).

Zdrojový kód obslužného programu je obsažen v souboru `Tuner_ATmega.c` (příloha B.1), hlavičkovém souboru `Tuner_ATmega.h` (příloha B.2) a knihovně funkcí `Tuner_ATmega_lib.c` (příloha B.3).

4.1 Spínání relé

Pro sepnutí jednotlivých relé je potřeba poslat vybranou kombinaci LC sériovou komunikací posuvným registrům s výkonovými spínači. Komunikaci obstarává funkce `relay()`, jejíž vstupní hodnotou je šestnáctibitová kombinace (viz. tab. 4.1). Logická jednička značí sepnuté relé, logická nula relé rozepnuté. Funkce postupně vystavuje hodnotu bitů kombinace od LSB (Least Significant Bit) k MSB (Most Significant Bit) na pinu DATA a posuvné registry ji čtou s kladným impulzem na pinu SCK (Shift Register Clock). Registry jsou sériově spojeny tak, že vytváří jeden spojitý šestnáctibitový registr. Po naplnění obou registrů je změna stavu příslušných spínačů provedena kladným pulzem na pinu RCK (Register Clock). Posuvné registry pracují velmi rychle, reagují na řídící signály v řádu desítek megahertz a proto je celá funkce spínání relé vykonána za dobu přibližně 10 μs. Samotná relé pak změní svůj stav do 10 ms. Nastavení pinů mikrokontroléru pro komunikaci s posuvnými registry (DATA, SCK a RCK) jako výstupní a jejich uvedení do stavu logické nuly provádí funkce `tpic_init()`. Funkce je volána jen jednou a to po zapnutí anténního tuneru.

Tab. 4.1: Šestnáctibitová kombinace určující stav sepnutí relé

MSB															LSB
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L0	L1	L2	L3	L4	L5	L6	-	C0	C1	C2	C3	C4	C5	C6	SW

Pro vytvoření kombinace (viz. tab. 4.1) slouží funkce `create_comb()`. Jejími parametry jsou sedmibitové hodnoty vyjadřující vybrané cívky (`coil`) a kondenzátory (`capacitor`) a jeden bit určující zvolenou topologii článku (`network`). Logické uspořádání parametrů určuje tab. 4.2. Uspořádání respektuje hodnoty indukčností a kapacit jednotlivých prvků a uspořádává je od nejmenší hodnoty po největší (od LSB k MSB). Výsledná šestnáctibitová kombinace je ovšem uspořádána v opačném smyslu. Je to dáno fyzickým rozložením relé na desce plošných spojů. Při vytváření kombinace

tak nestačí vstupní parametry jen bitově posunout a sečíst, ale je třeba přerovnat bity uspořádané od LSB k MSB na uspořádání od MSB k LSB. Z tohoto důvodu byla vytvořena funkce `swap_7_bits()`, která přerovnává sedmibitové hodnoty popsaným způsobem.

Tab. 4.2: Parametry funkce `create_comb()`

	MSB						LSB
	6	5	4	3	2	1	0
coil	L6	L5	L4	L3	L2	L1	L0
capacitor	C6	C5	C4	C3	C2	C1	C0
network	-	-	-	-	-	-	SW

4.2 Externí paměť EEPROM

Pro uložení kombinací LC pro pozdější použití je tuner vybaven pamětí EEPROM (viz. kapitola 3.3). Paměť má vnitřní uspořádání 32768 x 8 bitů. Jeden záznam šestnáctibitové kombinace zabírá dvě paměťová místa. Každá kombinace LC odpovídá vždy kmitočtu, na němž bylo přizpůsobení antény provedeno. Proto je na základě hodnoty kmitočtu vypočteno místo v paměti, na kterém bude kombinace uložena. Při použití stejného nebo podobného kmitočtu v budoucnu bude stejným způsobem určeno konkrétní paměťové místo s odpovídající kombinací a ta bude načtena. Záznamy jsou prováděny pro kmitočty 0 Hz až 32 MHz s krokem 4 kHz. Příklad obsahu paměti zobrazuje tab. 4.3. Celkem má paměť 8000 záznamů.

Tab. 4.3: Ukázka obsahu externí paměti EEPROM

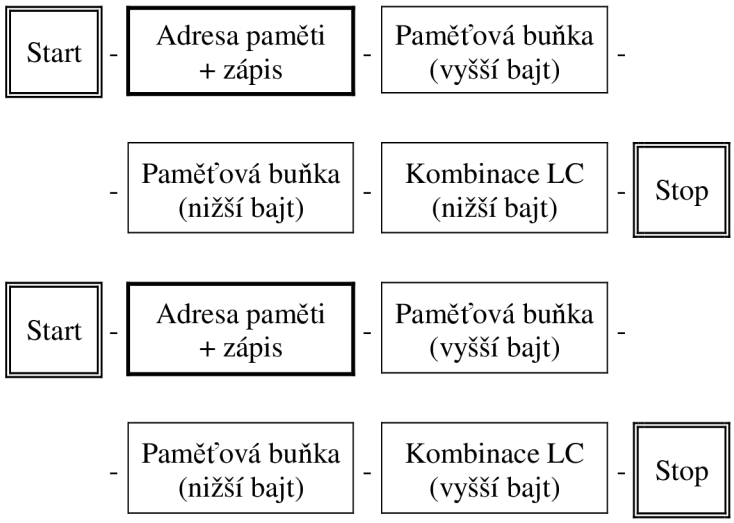
Kmitočet [kHz]	Paměťové místo	MSB							LSB
		7	6	5	4	3	2	1	0
3200	1600	C0	C1	C2	C3	C4	C5	C6	SW
	1601	L0	L1	L2	L3	L4	L5	L6	-
3204	1602	C0	C1	C2	C3	C4	C5	C6	SW
	1603	L0	L1	L2	L3	L4	L5	L6	-
3208	1604	C0	C1	C2	C3	C4	C5	C6	SW
	1605	L0	L1	L2	L3	L4	L5	L6	-
3212	1606	C0	C1	C2	C3	C4	C5	C6	SW
	1607	L0	L1	L2	L3	L4	L5	L6	-

Pro ukládání a načítání záznamů z paměti slouží funkce `save_comb()` a `load_comb()`. Obě funkce využívají pro komunikaci po sběrnici I^2C knihovnu funkcí `twimaster` (viz. [19]). Knihovna obsahuje funkce obsluhující řadič sběrnice I^2C zabudovaný v mikrokontroléru.

Sběrnice I^2C, označovaná firmou Atmel jako TWI (Two-Wire serial Interface), se skládá ze dvou vodičů – SDA (Synchronous Data) a SCL (Synchronous Clock).

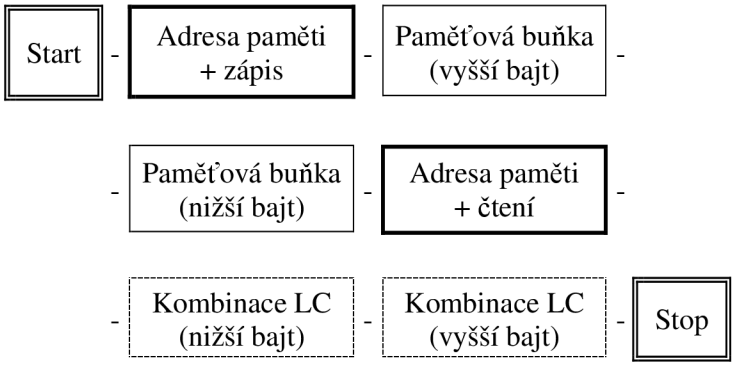
Mikrokontrolér se chová na sběrnici jako řídící zařízení (master) a paměť jako zařízení podřízené (slave). Slave přijímá od masteru pokyny, je řízen hodinovým signálem SCL a odpovídá jen, když je k tomu vyzván. Samotná komunikace probíhá pomocí paketů. Každý paket začíná tzv. startovací podmínkou a končí tzv. ukončovací podmínkou, při kterých dochází ke změně úrovně signálu na vodiči SDA v době, kdy je hodinový signál SCL ve vysoké úrovni. První je zasílán paket adresní, následuje jeden nebo více paketů datových. Všechny pakety jsou devítibitové. Adresní paket obsahuje sedmibitovou adresu podřízeného zařízení, bit určující, zda se bude do zařízení zapisovat nebo se z něho bude číst, a potvrzovací bit ACK (Acknowledge). Potvrzovací bit je generován podřízeným zařízením, které tak dává najevo masteru, že rozpoznalo svoji adresu a je připraveno ke komunikaci. Za adresním paketem následuje paket datový obsahující osmibitový příkaz pro podřízené zařízení, které potvrzuje přijetí generováním devátého bitu ACK. Další datový paket je obvykle generován podřízeným zařízením, které bylo příkazem vyzváno k zaslání dat. Master může po přijetí osmi datových bitů generovat bit ACK a vyžádat si tak další datový paket od podřízeného zařízení nebo může komunikaci standardně ukončit ukončovací podmínkou.

V případě komunikace s pamětí EEPROM je po zapnutí tuneru volána funkce `i2c_init()`, která nastavuje kmitočet hodinového signálu SCL na hodnotu 100 kHz. Zápis do paměti (funkce `save_comb()`) probíhá vysláním adresního paketu s adresou paměti EEPROM (adresa 0xA0) a bitem určujícím zápis pomocí funkce `i2c_start_wait()`. Funkce čeká na potvrzení ACK od paměti, které může být generováno s určitým zpožděním, pokud ještě není paměť na komunikaci připravena a probíhá v ní například zápis nařízený předcházejícím příkazem. V následujících dvou datových paketech zasílá funkce `i2c_write()` pozici v paměti, na kterou se bude zapisovat. Třetím datovým paketem se do paměti přenese nižších osm bitů zapisované šestnáctibitové kombinace prvků LC, opět pomocí funkce `i2c_write()`. Komunikace je ukončena ukončovací podmínkou, kterou vytváří funkce `i2c_stop()`. Uvedeným postupem je do paměti zapsána polovina kombinace. Celý postup je proto opakován s tím rozdílem, že je hodnota zasílané adresy o jednu pozici vyšší a ve třetím datovém paketu se zasílá vyšších osm bitů ukládané kombinace LC.



Obr. 4.1: Znázornění komunikace I^2C při zápisu do paměti EEPROM

Načtení kombinace z externí paměti provádí funkce `load_comb()`. Paměti je zaslán adresní paket s požadavkem na zápis. Následují dva datové pakety určující pozici v paměti pro čtení. Externí paměti je znovu zaslán adresní paket, tentokrát ale s požadavkem na čtení. Paměť EEPROM v následujícím datovém paketu posílá vyžádanou hodnotu (nižších osm bitů kombinace prvků LC). Paměť umožňuje čtení následujícího paměťového místa pouhým potvrzením ACK od masteru. K tomu je použita funkce `i2c_readAck()`. Paměť po potvrzení zasílá vyšších osm bitů uložené kombinace prvků. Master je čte a nevyžaduje čtení dalších dat pomocí funkce `i2c_readNak()`. Následuje ukončovací podmínka.



Obr. 4.2: Znázornění komunikace I^2C při čtení z paměti EEPROM

Časové průběhy komunikace I^2C mezi mikrokontrolérem a pamětí EEPROM představují obr. 4.1 a obr. 4.2. Adresní pakety jsou orámovány tučnou čarou, datové pakety vysílané masterem čarou jednoduchou a data vysílaná podřízeným zařízením čarou přerušovanou.

4.3 Kmitočtový čítač

Pro měření kmitočtu vysokofrekvenčního signálu na přizpůsobovaném vedení je použita srovnávací metoda, při které se porovnává kmitočet dvou čítačů/časovačů obsažených v mikrokontroléru. Obvod čítače (viz. kapitola 3.4) upravuje signál z vedení pro čítač 0 mikrokontroléru. Aby byl splněn vzorkovací teorém, musí být snímaný kmitočet více jak dvakrát menší než kmitočet vzorkovací daný hodinovým signálem mikrokontroléru (16 MHz). Proto je kmitočet vstupního signálu dělen v obvodu čítače čtyřmi. Teoreticky je tak možné dosáhnout měření kmitočtu do hodnoty 32 MHz, což pokrývá celé pásmo krátkých vln, pro které je tuner navržen.

Měření kmitočtu provádí funkce `get_frequency()`. Čítač/časovač 0 počítá vstupní pulzy. Po době dané periodou přetečení časovače 2 řízeného hodinovým signálem mikrokontroléru je přečtena aktuální hodnota pulzů N časovače 0 a pomocí vzorce (4.1) je určen kmitočet vstupního signálu f. Pro zajištění větší přesnosti měření je uvedený postup několikanásobně opakován a výsledek se průměruje.

$$f = 4 \cdot N \cdot f_2 \tag{4.1}$$

Hodnota f_2 odpovídá kmitočtu přetečení časovače 2 určeným pomocí výpočtu (4.2) z hodnoty hodinového kmitočtu mikrokontroléru f_{CLK}, předděličky kmitočtu M a počtu bitů B registru časovače 2. Přesná hodnota f_2 respektující prodlevu způsobenou vykonáváním obslužného programu byla na základě měření upravena tak, aby byl měřený kmitočet f co nejpřesněji změřen.

$$f_2 = \frac{f_{\mathrm{CLK}}}{\mathrm{M}\cdot 2^{\mathrm{B}}} = \frac{16\cdot 10^6}{1024\cdot 2^8} = 61{,}04\ \mathrm{Hz} \tag{4.2}$$

4.4 Měření poměru stojatých vln

Poměr stojatých vln (PSV) vypočítává mikrokontrolér na základě měření obvodem PSV-metru. PSV-metr poskytuje usměrněná napětí úměrná velikosti postupné a odražené vlny na vedení. Obě napětí jsou z analogové podoby převáděna analogově-digitálním (A/D) převodníkem na desetibitovou digitální hodnotu. Nastavení A/D převodníku provádí funkce `adc_init()`. Převodník by měl pracovat s hodinovým kmitočtem v rozsahu 50 kHz až 200 kHz. Proto je nastavena předdělička hodinového signálu mikrokontroléru na hodnotu 128 a A/D převodník tak pracuje s kmitočtem 125 kHz. Pro měření napětí na vybraném kanálu slouží funkce `getVoltage()`. Funkce provede pět A/D převodů napětí a z průměrné hodnoty vypočte dle vzorce (4.3) číselnou hodnotu napětí.

$$U = U_{\mathrm{digit}} \cdot \frac{U_{\mathrm{ref}}}{2^n - 1} = U_{\mathrm{digit}} \cdot \frac{2{,}5}{1023}\ [V] \tag{4.3}$$

U_{ref} symbolizuje referenční napětí 2,5 V, n je počet bitů A/D převodníku a U_{digit} reprezentuje digitálně vyjádřenou hodnotu napětí zapsanou v desítkové soustavě.

Měření a výpočet poměru stojatých vln provádí funkce `getSWR()`. Funkce měří napětí v obou kanálech (postupnou i odraženou vlnu), digitalizované napětí přepočítává na hodnotu napětí obdobně jako funkce `getVoltage()` s tím rozdílem, že výsledná napětí jsou průměrována z dvaceti naměřených hodnot. Průměrování je nutné pro potlačení šumu, který obzvláště při menších hodnotách měřeného napětí negativně ovlivňuje přesnost měření. Poměr stojatých vln je vypočten podle vzorce (1.5) a je limitován na hodnotě PSV = 999.0 z důvodu jednoduššího zobrazování informace v obslužném programu v osobním počítači.

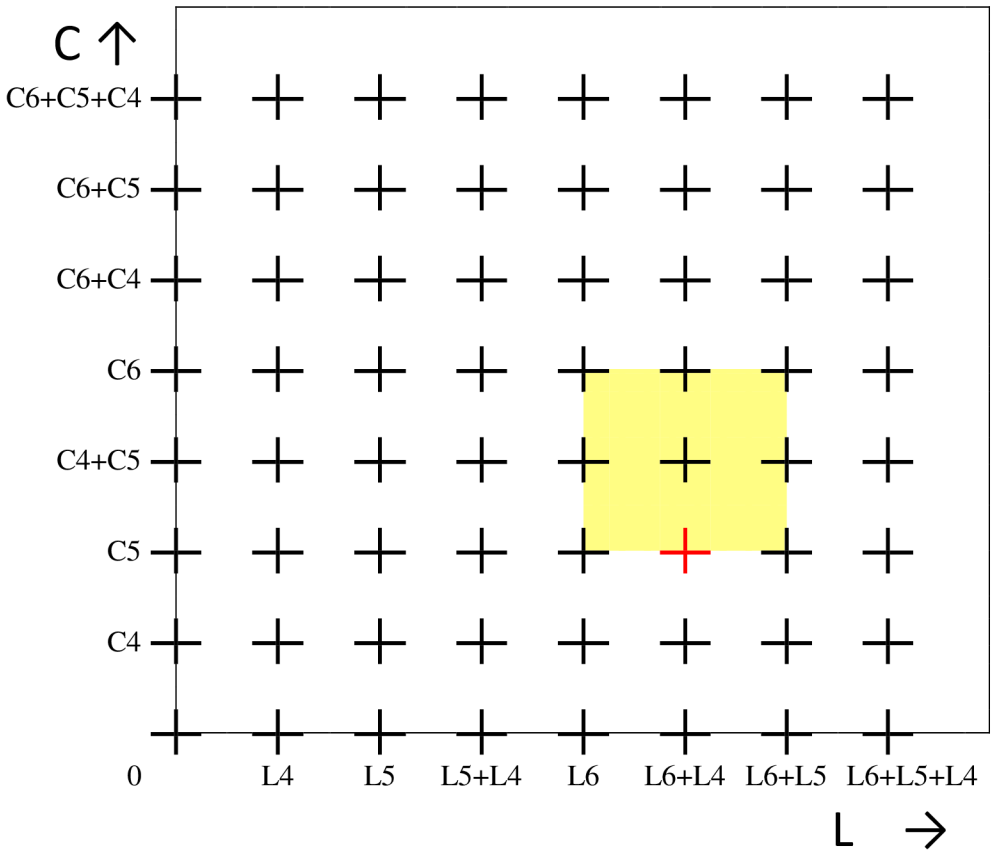
4.5 Inteligentní algoritmus ladění

Anténní tuner může být plně automatický jedině v případě, kdy obsahuje nějaký algoritmus na vyhledání správné kombinace LC pro přizpůsobení antény. Algoritmus posuzuje přizpůsobení na základě měřeného poměru stojatých vln. Jednoznačným postupem musí být vybrána jediná ze 32768 možných kombinací. Vybírá se ze 128 hodnot indukčnosti, 128 kapacity a možnosti měnit článek připnutím kapacity na stranu antény nebo na stranu vysílače. Pokud má L článek kapacitu připnutou na straně antény, jedná se o stranově otočený L článek, často označovaný jako článek Gama podle podobnosti tvaru s písmenem řecké abecedy Γ (velké gamma). Oba články se zásadně

liší v rozsahu možných přizpůsobovaných impedancí antény. L článek kvalitně přizpůsobuje antény s hodnotou impedance nižší než je požadovaná charakteristická impedance napáječe. Naopak Gama článek má schopnost dobře přizpůsobit zátěž s vyšší impedancí.

Inteligentní algoritmus, představovaný funkcí `Tuning()`, musí nejprve určit, zda má být použit článek L nebo Gama, a následujícími kroky vybrat nejvýhodnější kombinaci LC. Pro tento účel byl použit mechanismus vyhodnocování PSV pro velmi odlišné kombinace LC s následným upřesněním jednotlivých parametrů L a C pro kombinaci jevící se jako nejvýhodnější.

Ladění probíhá ve třech etapách. V první části jsou použity tří největší prvky indukčnosti a tří největší prvky kapacity na desce plošných spojů. Každé tři prvky mohou vytvořit až osm hodnot daného parametru. Možných kombinací indukčnosti a kapacity je tak celkem 64 a s uvážením použití dvou článků je výsledných možností 128. Kombinace jsou jedna za druhou spínány. Pro každý výběr prvků je změřen poměr stojatých vln. Mezi pokynem pro sepnutí relé a započetím měření PSV je prodleva v řádu desítek milisekund respektující dobu, kterou potřebují relé pro sepnutí, a také přechodové jevy vznikající připojováním indukčnosti a kapacity k vedení. Z měřených hodnot PSV je vybrána ta nejnižší. Odpovídající kombinace LC určuje, který článek bude zvolen a zároveň stanoví, kolem jakých hodnot kapacity a indukčnosti má algoritmus dále zpřesňovat svůj výběr. Daný postup zobrazuje obr. 4.3 formou



Obr. 4.3: Znázornění první etapy výběru kombinace LC

zobrazení kombinací LC jako bodů v rovině L-C. Červený bod reprezentuje kombinaci s nejnižším PSV a přilehlá vyznačená oblast představuje množinu kombinací LC, na které se algoritmus zaměří v dalších krocích.

Druhá etapa využívá třetích, čtvrtých a pátých nejvyšších hodnot prvků L a C. Z nich utvoří 64 kombinací, které v pomyslné rovině L-C rovnoměrně pokrývají oblast okolo kombinace vybrané v první etapě. První dva prvky L a C určené v první etapě jsou již neměnné. Třetí nejvyšší prvek proměnným zůstává. Všechny sestavené kombinace jsou opět postupně vybrány pomocí relé a pro každou je změřen PSV. Vybrána je kombinace s nejlepším (nejnižším) PSV. Průběh druhé etapy dokumentuje následující úryvek ze zdrojového kódu funkce `Tuning()`:

```
// 2. etapa
L_best = 0;         // indexy kombinace s nejlepším PSV
C_best = 0;
SWR_best = 999.0;   // nejlepší dosažený poměr stojatých vln

for(C_index = 8; C_index>0; C_index--)
  for(L_index = 8; L_index>0; L_index--)
  {
        // zpoždění před sepnutím relé
        _delay_ms(DELAY_BEFORE_SWITCH);

        // vytvoření kombinace a sepnutí relé
        relay(create_comb(coil | (L_index-1)<<2,
              capacitor | (C_index-1)<<2, network));

        // zpoždění po sepnutí relé
        _delay_ms(DELAY_AFTER_SWITCH);

        SWR = getSWR();   // měřeni poměru stojatých vln

        if(SWR<=SWR_best) // porovnání s nejlepším PSV
        {
        // náhrada předchozí kombinace s nejlepším PSV
        // nově zjištěnou kombinací s nižším PSV
              SWR_best = SWR;
                    L_best = L_index-1;
              C_best = C_index-1;
        }
  }
// uložení 3. a 4. největšího prvku L a C pro použití v 3. etapě
coil = coil | (L_best & 0x06)<<3;
capacitor = capacitor | (C_best & 0x06)<<3;
```

Třetí poslední etapa funguje obdobně jako etapa druhá. Čtyři největší prvky L a C jsou neměnné. Probíhají variace tří nejmenších prvků, což zahrnuje 64 kombinací. Výsledkem je kombinace s nejnižším PSV. Algoritmus ladění splnil svou roli a předává kompletní šestnáctibitovou kombinaci LC pro další zpracování jinými funkcemi, např. pro uložení do paměti EEPROM. Průběh celého algoritmu není, i přes velké množství spínaných kombinací LC, příliš časově náročný. Ladění trvá přibližně patnáct sekund.

4.6 Komunikace s převodníkem USB/UART

Ovládání anténního tuneru pomocí sběrnice USB zprostředkovává USB/UART převodník FT232R. Převodník komunikuje s mikrokontrolérem sériovým rozhraním UART.

Standard UART využívá asynchronní komunikace, kdy není nutné, aby vysílač vysílal data v pevně daných časových okamžicích. Prodleva mezi jednotlivými vysíláními je proměnná a závisí pouze na aktuální potřebě přenášet data. Hodinový signál přijímače nemusí být s hodinovým signálem vysílače synchronizován.

Data jsou přenášena v rámcích s předem daným formátem. Každý rámec obsahuje start bit, pět až devět datových bitů, volitelný paritní bit a jeden nebo dva stop bity. Vodič, po kterém je signál přenášen, je tažen pull-up rezistorem k vysoké úrovni (většinou 5 V). V klidovém stavu (bez vysílání) tak přijímač detekuje logickou jedničku. Start bit, který je reprezentován nízkou úrovní (logickou nulou), způsobí, že se přijímač zasynchronizuje na přijímanou sekvenci dat. Následují datové bity ve tvaru logická jednička = vysoká úroveň, logická nula = nízká úroveň. Paritní bit slouží k provedení jednoduché kontroly, zda nejsou přijatá data porušena. Při sudé paritě například doplňuje počet logických jedniček ve vysílaném rámci na sudý počet. Pokud přijímač přijme lichý počet logických jedniček, rozezná tímto způsobem chybu v přijímaných datech. Zabezpečení paritou je ovšem funkční jen pro jednonásobné chyby. Když přijímač přijme datové bity se dvěma chybami stejného charakteru (např. dvě log. 0 se změnily na dvě log. 1), vyhodnotí počet logických jedniček jako sudý a považuje přijatá data za bezchybná. Přenášený rámec ukončuje stop bit (log. 1), za kterým může následovat další datový rámec.

Pro správnou reprezentaci přijímaných dat musí přijímač znát formát vysílaných rámců a musí být nastaven na stejnou symbolovou rychlost jako vysílač. Symbolová rychlost SR v Baudech [Bd] určuje, jak rychle se mění logická úroveň posílaných dat (viz. (4.4)). Přijímač podle ní volí, v jakých okamžicích má snímat datový vodič. Pokud je symbolová rychlost nastavena chybně, například jako dvojnásobná, přijímač vyhodnotí každý příchozí symbol (logickou úroveň) jako dva stejné po sobě jdoucí symboly.

$$SR[Bd] = \frac{1}{\check{c}as\ mezi\ p\check{r}echody\ [s]} \tag{4.4}$$

Mikrokontrolér ATmega16 je vybaven tzv. jednotkou USART (Universal Synchronous Asynchronous serial Receiver and Transmitter), která obsahuje jeden přijímač a jeden vysílač UART. Převodník FT232R obsahuje také po jednom vysílači a přijímači. Spojením obou obvodů dvěma signálovými vodiči vzniká možnost duplexní (současně obousměrné) komunikace. Pro přenos dat byl zvolen formát rámců 8N1 (viz. tab. 4.4) a symbolová rychlost 9600 Bd.

Tab. 4.4: Formát rámce dat 8N1

Start bit	LSB			8 datových bitů				MSB	Stop bit
	0	1	2	3	4	5	6	7	

Obsluha jednotky USART je programově řešena pomocí funkcí knihovny `uart` (viz. [20]). Po zapnutí tuneru je volána funkce `uart_init()`, kterou je jednotka USART zapnuta, je nastavena symbolová rychlost a formát komunikace. Knihovna funkcí využívá možnost jednotky USART generovat přerušení (požadavek na obsluhu programem) při příjmu nebo při odeslání jednoho rámce dat. Rámec je při přijetí vyzvednut z paměti přijímače a uložen do zásobníku. Přijímač cyklicky přijímá rámce a přepisuje svou paměť vždy novým rámcem. Nevyzvednutý rámec je tak příjmem nově příchozího rámce ztracen. Zásobník rámců je programově vytvořen i pro přijímač, který s každým odeslaným rámcem zásobník postupně vyprazdňuje.

Formát osmi datových bitů v jednom rámci je výhodný z hlediska programování a odpovídá v jazyce C datovému typu `char` (znak). Pro příjem znaku (rámce) slouží funkce `uart_getc()`, která čte data ze zásobníku přijímače. Opakem je funkce `uart_putc()` pro zapsání vysílaného znaku do zásobníku vysílače. Pokud je potřeba vyslat více než jen jeden znak, je možné využít funkce `uart_puts()`, která posílá řetězec znaků.

Tab. 4.5: Seznam možných chyb při příjmu znaku sériovou komunikací UART

Název chyby	Popis
UART_FRAME_ERROR	Chyba rámce: nebyl detekován stop bit
UART_OVERRUN_ERROR	Paměť přijímače není čtena dostatečně rychle, byl ztracen příchozí paket
UART_BUFFER_OVERFLOW	Zásobník přijímače je plný, nebyl do něj zapsán příchozí paket
UART_NO_DATA	Zásobník přijímače je prázdný

Knihovna funkcí `uart` ve spolupráci s jednotkou USART umožňuje detekovat chyby nastalé při příjmu dat. Jednotlivé chyby dokumentuje tab. 4.5. Tyto chyby jsou číselně vyjádřeny a je možné je číst pomocí funkce `uart_getc()`. Funkce vrací dvoubajtovou hodnotu. V nižším bajtu se nachází přijatý znak a ve vyšším kód chyby, který dále zpracovává funkce `UartError()`.

4.7 Vzdálené řízení

Obslužná aplikace v osobním počítači (PC) sloužící ke vzdálenému řízení anténního tuneru je při spojení s tunerem v nadřazené pozici. Vydává příkazy, na které tuner reaguje. Při komunikaci je nutné zasílat datové znaky dle určitého protokolu. Požadavkem je, aby byla přenášená informace na obou stranách komunikačního řetězce jednoznačně identifikována a byla provedena odpovídající reakce. Z tohoto důvodu musí být, kromě formátu komunikace, také pevně určen počet zasílaných znaků. Příkladem může být požadavek obslužné aplikace na měření kmitočtu signálu na přizpůsobovaném vedení. Aplikace vyšle znak odpovídající této činnosti. Tuner znak rozpozná, provede měření kmitočtu a hodnotu kmitočtu odešle aplikaci zpět pomocí předem daného počtu znaků. Pokud by tuner odeslal méně znaků, nebylo by na straně obslužné aplikace možné informaci o kmitočtu identifikovat. Pokud by odeslal znaků

více, aplikace by přijala jen očekávaný počet znaků. Přebytečné znaky by pak čekaly ve virtuálním zásobníku portu v PC. Při příští komunikaci, kdy by chtěla obslužná aplikace přijímat od tuneru data, by se načetly nejprve dříve přijaté přebytečné znaky a až poté znaky nově příchozí. Aplikace by tak získala jiná data, než která požadovala, což by mohlo vést až k chybě znemožňující aplikaci další činnost.

Logický protokol pro přenášené znaky zahrnuje jak kontrolu přijatých dat v tuneru, tak i zpětnou kontrolu vyslaného příkazu samotnou aplikací v PC. Na straně tuneru je komunikace řízena funkcí `RemoteControl()`. Komunikace začíná vždy tak, že aplikace v PC vyšle znak reprezentující určitý příkaz. Za prvním znakem mohou následovat další v závislosti na povaze vysílaného příkazu. Funkce `RemoteControl()` je v programu mikrokontroléru periodicky volána a zjišťuje, zda byl přijat znak. Pokud ano, je funkcí `UartError()` vyhodnoceno, zda nenastala při příjmu chyba. Bezchybně přijatý znak je odeslán aplikaci v PC zpět pro kontrolu, zda nedošlo k chybě v přenosovém kanálu (zda není číselná hodnota přijatého znaku odlišná od hodnoty vysílané). Znak, u něhož byla vyhodnocena chyba příjmu na straně tuneru, je nahrazen znakem 1 a odeslán obslužné aplikaci následován znakem reprezentujícím nastalou chybu (viz. tab. 4.5). Tímto způsobem je možné o chybě informovat uživatele obslužné aplikace.

Tab. 4.6: Seznam příkazů zasílaných obslužnou aplikací v PC

Příkaz	Přijatých /odeslaných znaků	Popis příkazu
1	1/1	Připojení vzdáleného řízení
2	1/1	Odpojení vzdáleného řízení
10	1/3	Zaslat hodnotu kombinace LC
20	3/1	Sepnout relé odpovídající zasílané kombinaci LC
30	1/12	Zaslat napětí na kanálu 1 A/D převodníku
31	1/12	Zaslat napětí na kanálu 0 A/D převodníku
32	1/12	Zaslat hodnotu poměru stojatých vln
33	1/1	Zapnout/vypnout pull-up rezistory na vstupních pinech A/D převodníku
40	1/5	Zaslat hodnotu kmitočtu
50	1/1	Spustit algoritmus automatického ladění
60	1/1	Uložit kombinaci LC do paměti EEPROM

Tuner na vyžádání příslušným příkazem zasílá aplikaci různá data ve formě znaků následujících první znak s informací o příkazu. Šestnáctibitová kombinace LC je zasílána jak aplikací tak i tunerem ve formě dvou znaků. První nese horních osm bitů kombinace a druhý dolních osm bitů. Kmitočet je vyjádřen dvaatřicetibitovým celým číslem a je proto přenášen čtyřmi znaky. V případě zasílání hodnoty měřeného napětí nebo poměru stojatých vln, které jsou vyjádřeny 64-bitovým číslem s plovoucí řádovou čárkou, nelze číslo jednoduchými operacemi (např. bitovým posunem) rozdělit na osmibitové znaky. Je proto využita funkce `dtostre()` z knihovny funkcí `stdlib`. Funkce převede 64-bitovou hodnotu na text ve tvaru ±*d.dddd*E±*dd*, kde *d* představuje

číslice a E reprezentuje násobení číslem deset umocněným na následující dvoumístnou hodnotu. Celý text je vyjádřen 11 znaky, které se přenáší obslužné aplikaci.

Význam jednotlivých příkazů a komunikační protokol s definováním počtu tunerem přijatých a odeslaných znaků popisuje tab. 4.6.

4.8 Automatický mód anténního tuneru

V případě, že není anténní tuner ovládán dálkově, nachází se v tzv. automatickém módu. Po zapnutí napájení je nastavena komunikace s jednotlivými obvody tuneru a následně dojde ke spuštění automatického módu. Programově je činnost tuneru řízena funkcí `main()`.

Funkce `main()` nejprve nastaví komunikaci s indikačním panelem LED. Je rožnuta zelená dioda jako indikace zapnutého napájení. Inicializuje se spojení s posuvnými registry TPIC a je zadána výchozí kombinace LC představující rozepnutí všech relé. Dále se nastaví komunikace s pamětí EEPROM. Je zapnut A/D převodník a nastaveny jeho parametry. Nakonec se provede inicializace sériové komunikace UART. Uvedené činnosti jsou prováděny odpovídajícími funkcemi. Jejich volání ukazuje následující úryvek ze zdrojového kódu funkce `main()`:

```
led_init();
PORTA &= ~_BV(GREEN);

tpic_init();
Kombinace = 0x0000;
relay(Kombinace);


i2c_init();
adc_init();
uart_init( UART_BAUD_SELECT(UART_BAUD_RATE,F_CPU) );
```

V automatickém módu setrvává program mikrokontroléru v nekonečné čekací smyčce. Měření poměrů na přizpůsobovaném vedení probíhá v obsluze přetečení časovače 1, ke které dochází přibližně jednou za sekundu. Obsluha přerušení je indikována blikáním žluté diody indikačního panelu. V první fázi, kdy tunerem neprochází užitečný signál, je přerušení využíváno k detekci napětí postupné vlny. Při nalezení užitečného signálu je změřen jeho kmitočet a je porovnáno, zda odpovídá kmitočtovému pásmu, pro které je tuner navržen. S úspěchem porovnání je zahájen proces ladění (indikace červenou diodou). Z paměti EEPROM je načtena kombinace LC, která byla pro daný kmitočet v minulosti naladěna. Je změřen poměr stojatých vln pro danou kombinaci a na jeho základě je rozhodnuto o nutnosti vyhledání nové kombinace LC. Vyhledání kombinace obstarává inteligentní algoritmus ladění, jehož výsledek je uložen do paměti EEPROM pro příští použití. Vyhledaná kombinace je sepnuta a je změřen aktuální poměr stojatých vln. Program se uvede do režimu vyčkávání. Při přerušení časovače 1 je tentokrát měřen poměr stojatých vln a je zjišťováno, zda nedošlo od okamžiku naladění k jeho velkému zhoršení. Případné nevyhovující PSV vyvolá opětovný proces přizpůsobování.

Funkce `main()` volá pravidelně funkci `RemoteControl()`. Zjišťuje se tak, zda

nebyl zaslán příkaz o připojení vzdáleného řízení. Při jeho příjmu je automatický režim tuneru zastaven a veškeré činnosti řídí obslužná aplikace v PC. Vzdálené řízení je signalizováno svitem žluté oznamovací diody. Vykonávání aktuálně zaslaného příkazu indikuje červená dioda. Tuner se vrací zpět do automatického režimu při příjmu příkazu „Odpojení vzdáleného řízení“.

5 OBSLUŽNÁ APLIKACE V OSOBNÍM POČÍTAČI

Dálkové ovládání tuneru a zobrazování měřených hodnot na přizpůsobovaném vedení zajišťuje obslužná aplikace v osobním počítači. Aplikace byla vytvořena v programovém prostředí Microsoft Visual C++ 2008 Express Edition (viz. [21]) pomocí programovacího jazyka C++. Bylo využito softwarové platformy Microsoft .NET Framework 3.5 (viz. [22]) a aplikační šablony Windows Forms Application. Zdrojový kód aplikace je překládán do tzv. mezijazyka Common Intermediate Language (CLI). Pro spuštění aplikace je vyžadováno prostředí Common Language Runtime (CLR). Uvedené technologie předurčují obslužnou aplikaci k použití v osobních počítačích s operačním systémem Microsoft Windows (viz. [23]).

5.1 Spojení s anténním tunerem

Obslužná aplikace umožňuje spojení s anténním tunerem pomocí dvou standardů komunikace – USB a RS-232. Tuner je standardně spojován s osobním počítačem pomocí datového kabelu USB A/B. Pokud je potřeba tuner ovládat ze vzdálenosti větší než přibližně pět metrů, je možné k tuneru připojit externí USB/RS-232 převodník (není součástí bakalářské práce). Externí převodník lze spojit se sériovým portem (RS-232) osobního počítače pomocí datového kabelu v délce okolo deseti metrů.

Pro spojení s tunerem přes USB bylo třeba nakonfigurovat USB/UART převodník FT232R a vytvořit ovladače zařízení, které vyžaduje operační systém. Do vnitřní paměti EEPROM převodníku FT232R byl pomocí programu FT_PROG 1.3.1 zapsán tzv. deskriptor zařízení (viz. [24]) nutný pro identifikaci zařízení. K deskriptoru byly programem FT_INF 1.0 vytvořeny odpovídající ovladače zařízení pro operační systém. Programy FT_PROG, FT_INF a podpůrné soubory ovladačů jsou poskytovány výrobcem převodníku FT232R – firmou FTDI (viz. [25]).

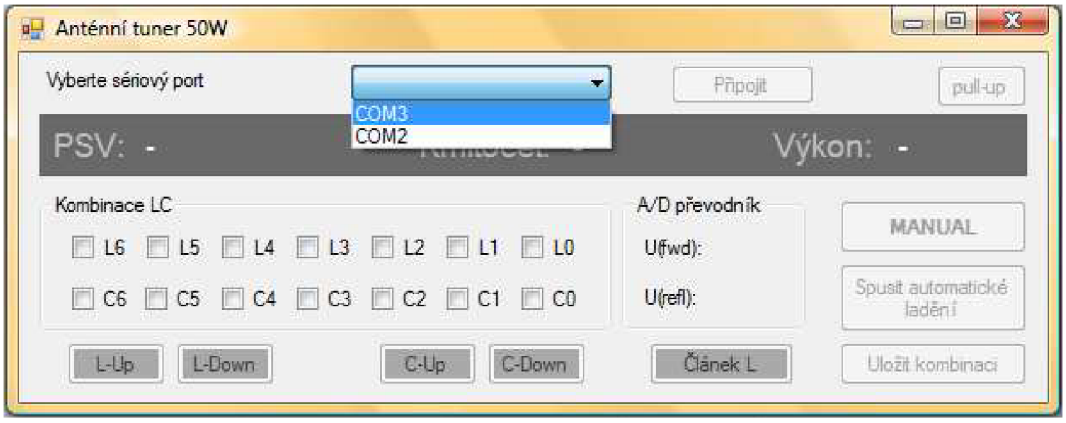
K instalaci ovladačů zařízení je uživatel vyzván operačním systémem při prvním připojení anténního tuneru. Ovladače je třeba vyhledat v adresáři počítače manuálně. Nejprve se zvolí soubor `ftdibus.inf`. Po jeho nainstalování a opětovné výzvě se vybere soubor `ftdiport.inf`. První ze souborů představuje ovladač pro USB port. Druhý vytváří tzv. virtuální sériový port. Umožňuje tak komunikaci s tunerem pomocí sériového portu, přestože je fyzicky spojen přes USB. Na tomto principu je založena činnost obslužné aplikace, která komunikuje s tunerem přes sériový port, fyzický (spojení RS-232) nebo virtuální (spojení USB).

Před prvním spuštěním obslužné aplikace je vhodné nastavit parametry komunikace sériového portu. Spustí se Správce zařízení systému Windows. Ze skupiny „Porty (COM a LPT)“ se vybere „Antenna Tuner (Serial Port)“ a dle tab. 5.1 se zvolí jeho vlastnosti. Vybere se možnost „Advanced…“ a v dialogu se nastaví parametr „COM Port Number“ na vysokou hodnotu, např. „COM20“. Nastavení se potvrdí tlačítkem „OK“. Vysoké číselné označení sériového portu je vhodné pro jednoznačnou identifikaci v obslužné aplikaci, kde se sestavuje komunikace s tunerem na základě výběru odpovídajícího portu.

Tab. 5.1: Nastavení formátu komunikace sériového portu

Parametr	Hodnota
Bits per second	9600
Data bits	8
Parity	None
Stop bits	1
Flow Control	None

Spojení obslužné aplikace s anténním tunerem provede uživatel následujícím postupem. Propojí datovým kabelem tuner a osobní počítač. Spustí obslužnou aplikaci (soubor `Antenna Tuner.exe`) a provede výběr portu, na kterém je tuner připojen (viz. obr. 5.1) a stiskne tlačítko „Připojit“.



Obr. 5.1: Výběr sériového portu pro komunikaci s anténním tunerem

5.2 Zobrazení měřených hodnot

Aplikace v osobním počítači automaticky zobrazuje hodnoty měřených veličin na přizpůsobovaném vedení. Hodnoty jsou pravidelně aktualizovány. Měří se poměr stojatých vln na vedení, kmitočet signálu, jeho výkon a přenáší se i hodnoty snímané A/D převodníkem, které jsou nezbytné například pro kalibraci PSV-metru. Měřené hodnoty zasílá tuner na základě příslušného příkazu (viz. tab. 4.6). Pokud je tunerem detekováno nízké nebo žádné napětí postupné vlny, aplikace namísto měřených hodnot zobrazí upozornění „žádný signál“.

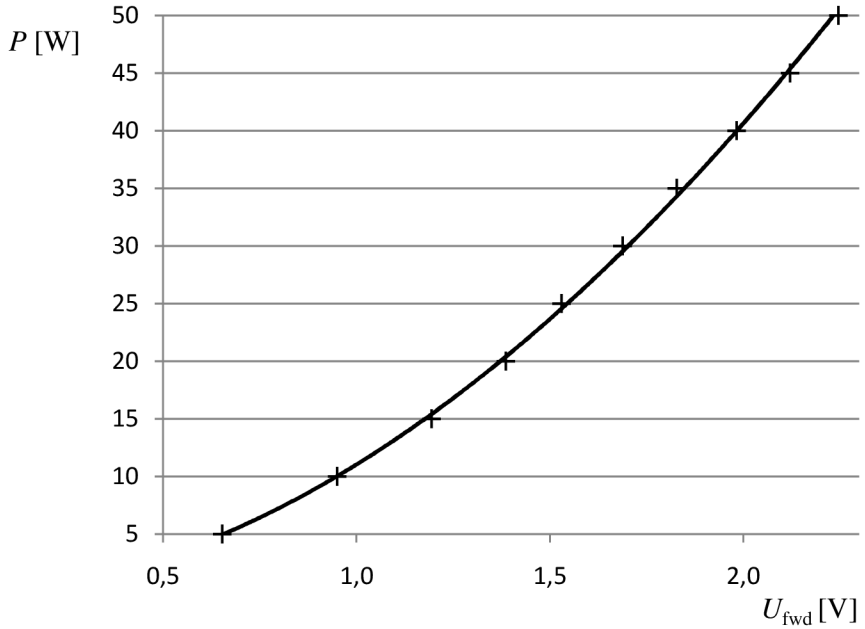
Měření výkonu signálu na vedení tuner neumožňuje přímo. Daná hodnota se určuje výpočtem z hodnoty napětí postupné vlny. Mezi napětím snímaným A/D převodníkem a napětím postupné vlny by měla být lineární závislost. Mezi napětím signálu a jeho výkonem platí vztah (5.1).

$$P = \frac{U^2}{R} \tag{5.1}$$

Na základě uvedených předpokladů bylo provedeno měření závislosti napětí postupné vlny na výkonu signálu. Při měření byl použit transceiver YAESU FT 950 a 50 Ω zátěž MFJ-250 (1 kW). Měření probíhalo na kmitočtu 14 MHz při PSV menším než 1,02. Naměřené hodnoty shrnuje tab. 5.2 a obr. 5.2.

Tab. 5.2: Závislost snímaného napětí a výkonu signálu

P [W]	U_{fwd} [V]	P [W]	U_{fwd} [V]
5	0,653	30	1,688
10	0,950	35	1,828
15	1,194	40	1,983
20	1,386	45	2,121
25	1,530	50	2,246



Obr. 5.2: Závislost snímaného napětí a výkonu signálu

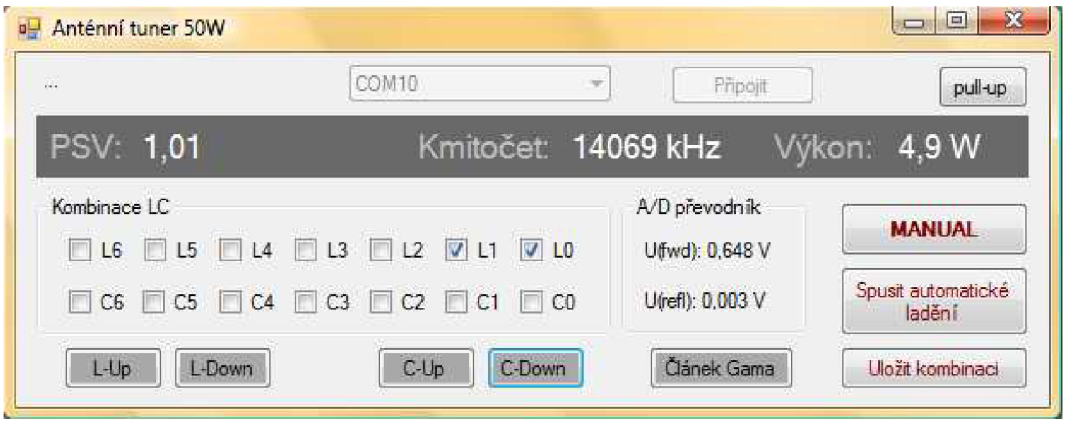
Výsledky byly proloženy křivkou s mocninnou závislostí $P = 11{,}034\ U_{fwd}^{\ 1{,}88}$, která je téměř kvadratická. Odchýlení křivky od předpokládané paraboly způsobila zřejmě menší nelinearita transformátoru v PSV-metru. Závislost je využita aplikací pro výpočet výkonu.

5.3 Režimy obslužné aplikace

Automatický režim anténního tuneru doplňuje obslužná aplikace o další dva módy. Jedná se o poloautomatický a manuální mód. Přepínání mezi módy umožňuje tlačítko „MANUAL/AUTO“.

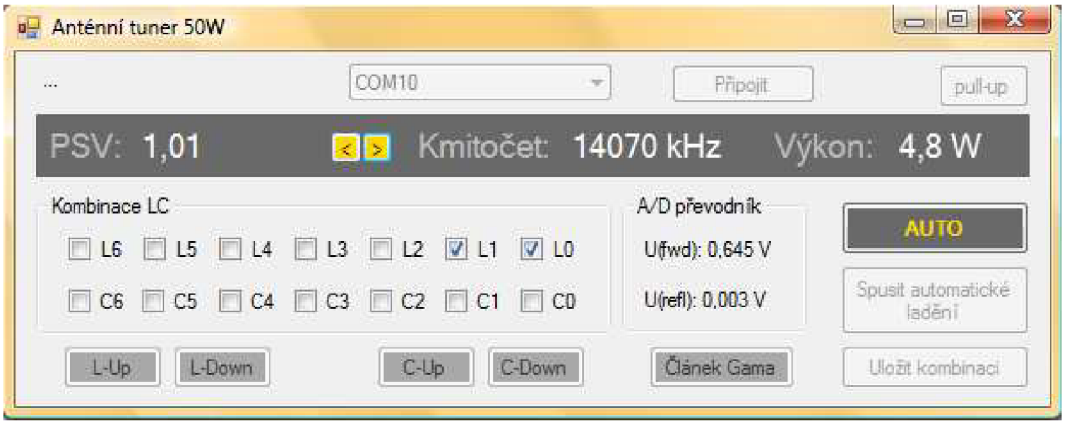
Manuální mód (viz. obr. 5.3) umožňuje uživateli spouštět přizpůsobovací

algoritmus tuneru dle potřeby (tlačítko „Spustit automatické ladění“). Kombinaci LC lze vybírat nebo upravovat pomocí tlačítek pro změnu hodnot indukčnosti „L-Up“, „L-Down“ a kapacity „C-Up“, „C-Down“. Změnu článku umožňuje přepínač „Článek L/ Článek Gama“. Kombinaci lze pro budoucí použití uložit tlačítkem „Uložit kombinaci“.



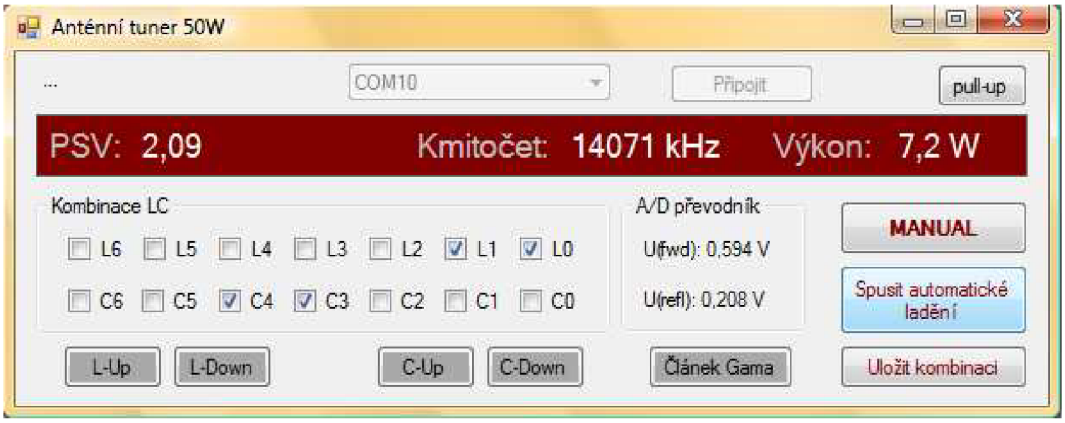
Obr. 5.3: Manuální mód obslužné aplikace

Poloautomatický mód, v aplikaci zjednodušeně označen jako „AUTO“ (viz. obr. 5.4), umožňuje nastavení cílového poměru stojatých vln. Aplikace sama hlídá, aby nebyl požadovaný PSV překročen, a rozhoduje o spuštění přizpůsobovacího algoritmu. PSV je možné nastavit žlutými tlačítky v rozmezí 1,1 – 3.



Obr. 5.4: Poloautomatický mód obslužné aplikace

Na probíhající ladění je obsluha tuneru upozorněna tmavočerveným podbarvením oznamovací části s měřenými hodnotami (viz. obr. 5.5).



Obr. 5.5: Indikace probíhajícího ladění

6 OVĚŘENÍ FUNKČNOSTI TUNERU

Za účelem kalibrace a kontroly přesnosti byly výsledky měření obvodů čítače a PSV-metru porovnány s profesionální měřicí technikou. Kmitočtový čítač byl srovnáván s vestavěným čítačem systému Metex Universal-System MS-9170. Při měření bylo zjištěno, že s rostoucím vstupním výkonem roste rozsah použitelnosti čítače směrem k nižším kmitočtům. Měření bylo prováděno při nízkém vstupním výkonu (100 mW) a proto by měly stanovené výsledky platit nebo se i zlepšovat pro vyšší hodnoty vstupního výkonu. Změřený rozsah použitelnosti čítače je 150 kHz – 30 MHz. Shora je rozsah omezen nutností splnění vzorkovacího teorému (viz. kapitola 4.3). Maximální odchylky změřeného kmitočtu udává tab. 6.1. Chyba měření se s hodnotou kmitočtu vstupního signálu zvyšuje.

Tab. 6.1: Maximální odchylky měření kmitočtu

Kmitočtový rozsah [kHz]	Δf_{max} [kHz]
150 – 1000	0,1
1000 – 5000	0,3
5000 – 10000	0,5
10000 – 30000	2

PSV-metr byl srovnán s měřením vestavěného PSV-metru transceiveru YAESU FT 950. Chyba měření poměru stojatých vln byla v celém pásmu krátkých vln menší než jedna desetina.

Po ověření vlastností měřicích obvodů anténního tuneru byla provedena zkouška schopnosti automaticky přizpůsobit impedanci antény. Generátorem signálu byl transceiver YAESU FT 950 a přizpůsobovanou zátěží anténa Vertical AVT4 určená pro pásma 7, 14, 21 a 28 MHz. Hodnoty poměru stojatých vln pro jednotlivé kmitočty signálu před a po přizpůsobení shrnuje tab. 6.2. Hodnoty PSV se ve většině případů zlepšily dostavením hodnot L a C v manuálním módu obslužné aplikace.

Tab. 6.2: Zkouška impedančního přizpůsobení antény Vertical AVT4 (P = 5 W)

Kmitočet [MHz]	PSV [-] před přizpůsobením	PSV [-] po přizpůsobení
7,00	3,80	1,90
10,10	5,90	1,90
14,07	1,92	1,48
21,10	4,80	1,08

7 ZÁVĚR

Po teoretickém rozboru impedančního přizpůsobení antény byly v práci uvedeny výhody a nevýhody různých impedančních transformátorů. S přihlédnutím k údajům o profesionálně vyráběných tunerech byla vybrána koncepce automatického anténního tuneru s přepínatelným L článkem. Pro zvolené řešení bylo vypracováno konkrétní obvodové zapojení. Byly navrženy dvě dvouvrstvé desky plošných spojů. Desky byly zhotoveny a osazeny součástkami. Pro tuner byla na míru vyrobena krabička z ohýbaného plechu. Byl vytvořen program v jazyce C pro řídící jednotku (mikrokontrolér ATmega) tuneru. Do programu byl zahrnut inteligentní algoritmus ladění. Tuner byl vybaven aplikací pro osobní počítač umožňující jeho dálkové ovládání.

Shrnutí elektrických i mechanických parametrů automatického anténního tuneru představuje tab. 7.1.

Tab. 7.1: Parametry automatického anténního tuneru

Kmitočtový rozsah	3 – 30 MHz
Vstupní výkon	max. 50 W, pro ladění doporučeno 5 W
Vstupní impedance	50 Ω
Rozsahy L a C	L: 0 – 10,3 μH (127 hodnot), C: 0 – 1270 pF (127 hodnot)
Typ článku	přepínatelný L – článek / Gama článek
Doba ladění	15 sekund
Paměťových pozic	8000
Zobrazení PSV	1,0 – 999,0
Zobrazení kmitočtu	150 – 30 000 kHz
Zobrazení výkonu	0 – 60 W
Napájecí napětí	12 V DC
Proudový odběr	max. 700 mA
Komunikační rozhraní	USB, RS-232 (s externím převodníkem)
Rozměry	172 mm (Š) x 200 mm (H) x 46 mm (V)

Navržený a sestavený tuner v celém pásmu krátkých vln spolehlivě měří poměr stojatých vln na vedení, kmitočet signálu i jeho výkon. Předpokládaná výkonová zatížitelnost 50 W byla ověřena a tuner při ní nevykazoval žádné odchylky od běžné činnosti. Inteligentní algoritmus zvládá automaticky přizpůsobovat zátěž na hodnotu činitele stojatých vln nižší než 2,0, v případě zásahu obsluhy nižší než 1,5. Všechny body zadání bakalářské práce byly splněny.

LITERATURA

[1] NOVÁČEK, Z. *Elektromagnetické vlny, antény a vedení Přednášky*. Brno: FEKT VUT v Brně, 2006.

[2] PROCHÁZKA, M. *Antény – Encyklopedická příručka*. Praha: BEN – technická literatura, 2000.

[3] DOSTÁL, T. *Elektrické filtry*. Brno: FEKT VUT v Brně, 2007.

[4] Antenna Tuner [online]. Wikimedia Foundation, Ltd. – [cit. 27. května 2009]. Dostupné na www: <http://en.wikipedia.org/wiki/Antenna_tuner.htm>

[5] SEDLÁČEK, J., VALSA, J. *Elektrotechnika II*. Brno: FEKT VUT v Brně, 2004.

[6] Elecraft KAT100 Automatic Antenna Tuner [online]. Aptos: Elecraft, 2003 – [cit. 8. září 2009]. Dostupné na www: <http://www.elecraft.com/manual/KAT100%20man%20rev%20C.pdf>

[7] Elecraft KAT2 Automatic Antenna Tuner: Assembly and Operating Instructions [online]. Aptos: Elecraft, 2001 – [cit. 8. října 2009]. Dostupné na www: <http://www.elecraft.com/manual/KAT2manRevD.pdf>

[8] Elecraft T1 Automatic Antenna Tuner: Owner's Manual [online]. Aptos: Elecraft, 2005 – [cit. 8. října 2009]. Dostupné na www: <http://www.elecraft.com/manual/T1%20owners%20man%20rev%20A2.pdf>

[9] IntelliTuner: Automatic Antenna Tuner [online]. Starkville: MFJ Enterprises, 2005 – [cit. 8. října 2009]. Dostupné na www: <http://www.mfjenterprises.com/man/pdf/MFJ-993.pdf>

[10] 8-bit Microcontroller ATmega16 [online] San Jose: Atmel Corporation, 2009 – [cit. 8. října 2009] Dostupné na www: <http://www.atmel.com/dyn/resources/prod_documents/doc2466.pdf>

[11] LUKEŠ, Z., ŠRÁMEK P. *Anténní tuner 1 kHz – 30 MHz*. Výzkumná zpráva projektu ST1880010. Brno: Ústav radioelektroniky FEKT VUT v Brně, 2010.

[12] FT232R USB UART I.C. [online]. Glasgow: Future Technology Devices International, 2005 – [cit. 20. října 2009]. Dostupné na www: <http://www.ftdichip.com/Documents/DataSheets/DS_FT232R_V205.pdf>

[13] L78M05 Positive Voltage Regulator [online]. STMicroelectronics, 2004 – [cit. 19. listopadu 2009]. Dostupné na www: <http://pdf1.alldatasheet.com/datasheet-pdf/view/22669/ STMICROELECTRONICS/L78M05.html>

[14] SILVERMAN, J.H. C Reference Card [online]. Providence: Brown University, 1999 – [cit. 16. května 2010]. Dostupné na www: <http://refcards.com/docs/silvermanj/ansi-c/ansi-c-refcard-a4.pdf>

[15] AVR Studio 4 Description [online]. San Jose: Atmel Corporation, 2010 – [cit. 16. května 2010]. Dostupné na www: <http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725>

[16] WEDDINGTON, E., SOKOLOV, A., WUNSCH, J., MARQUES, P. WinAVR [online]. 2010 – [cit. 19. května 2010]. Dostupné na www: <http://sourceforge.net/projects/winavr/>

[17] MICHALKIEWICZ, M., WUNSCH, J., WEDDINGTON, E., SOKOLOV, A., XMELKOV, D., avr-libc 1.6.7 [online]. 2010 – [cit. 19. května 2010]. Dostupné na www: <http://www.nongnu.org/avr-libc/>

[18] POVALAČ, A. AVR ISP Programátor BiProg – ÚREL verze [online]. Brno: Ústav radioelektroniky FEKT VUT v Brně, 2009 – [cit. 16. května 2010]. Dostupné na www: <http://www.urel.feec.vutbr.cz/web_documents/dilna/BiProg/biprog_urel.pdf>

[19] FLEURY, P. I2C (TWI) Master Software Library [online]. 2006 – [cit. 16. května 2010]. Dostupné na www: <http://hompepage.hispeed.ch/peterfleury/group__pfleury__i2cmaster.html>

[20] FLEURY, P. UART Library [online]. 2005 – [cit. 16. května 2010]. Dostupné na www: <http://hompepage.hispeed.ch/peterfleury/group__pfleury__uart.html>

[21] Microsoft Visual Studio Express [online]. Microsoft, 2010 – [cit. 23. května 2010]. Dostupné na www: <http://www.microsoft.com/cze/windows/default.aspx>

[22] Microsoft .NET Framework 3.5 [online]. Microsoft, 2010 – [cit. 23. května 2010]. Dostupné na www: <http://www.microsoft.com/downloads/details.aspx?familyid=333325fd-ae52-4e35-b531-508d977d32a6&displaylang=en>

[23] Microsoft Windows [online]. Praha: Microsoft, 2010 – [cit. 23. května 2010]. Dostupné na www: <http://www.microsoft.com/express>

[24] MATOUŠEK, D. *USB prakticky s obvody FTDI*. Praha: BEN – technická literatura, 2003.

[25] FTDI Utilities [online]. Glasgow: Future Technology Devices International, 2010 – [cit. 23. května 2010]. Dostupné na www: <http://www.ftdichip.com/Resources/Utilities.htm>

[26] MATOUŠEK, D. *Udělejte si z PC 1. díl*. Praha: BEN – technická literatura, 2002.

[27] MATOUŠEK, D. *Udělejte si z PC 2. díl*. Praha: BEN – technická literatura, 2002.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

$\vec{U}$	Napětí postupné elektromagnetické vlny
$\vec{I}$	Proud postupné elektromagnetické vlny
$\overleftarrow{U}$	Napětí odražené elektromagnetické vlny
$\overleftarrow{I}$	Proud odražené elektromagnetické vlny
Z_k	Impedance zátěže
Z_{ov}	Charakteristická impedance vedení
ρ	Činitel odrazu
σ	Poměr stojatých vln (PSV)
R	Odpor
L	Indukčnost
C	Kapacita
X	Reaktance
P	Výkon
f	Kmitočet
SR	Symbolová rychlost

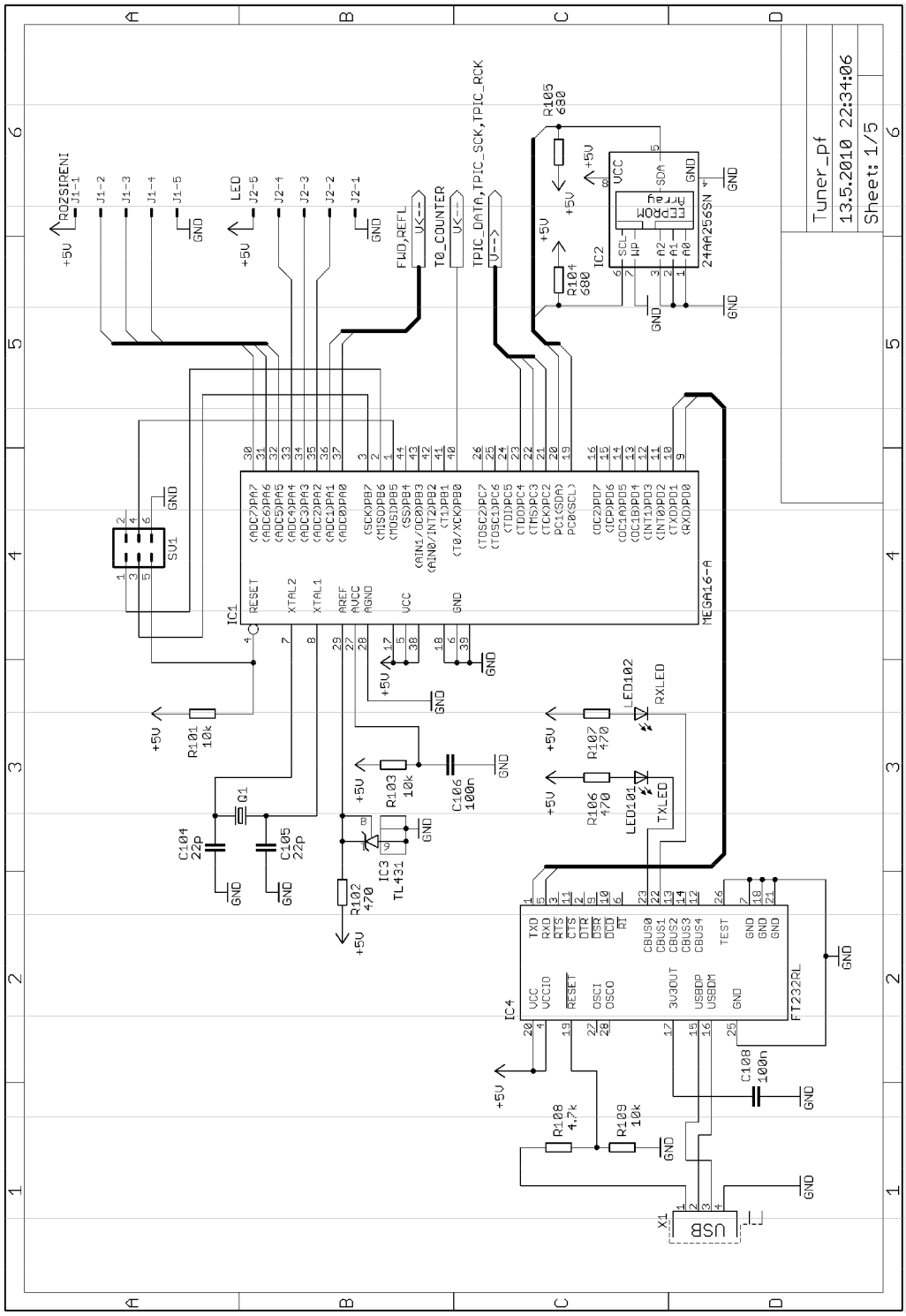
SWR	Standing Wave Ratio, poměr stojatých vln
USB	Universal Serial Bus, univerzální sériová sběrnice
A/D	Analogově/digitální
SPI	Serial Peripheral Interface, sériové rozhraní pro periferie
LED	Light Emitting Diode, světelná dioda
EEPROM	Electrically Erasable Programmable Read-Only Memory, elektricky mazatelná programovatelná paměť určená pro čtení
I^2C	Inter Integrated Circuit, komunikační sběrnice mezi integrovanými obvody
UART	Universal Asynchronous Receiver Transmitter, univerzální asynchronní přijímač / vysílač
ISP	In-System Programming, programování mikrokontroléru na desce plošných spojů, kde plní svou funkci, ne v externím programátoru
LSB	Least Significant Bit, nejméně významný bit

MSB	Most Significant Bit, nejvýznamnější bit
SCK	Shift Register Clock, hodinový signál posuvného registru
RCK	Register Clock, hodinový signál registru
TWI	Two-Wire serial Interface, sériová sběrnice sestávající ze dvou vodičů
SDA	Synchronous Data, synchronní datový signál
SCL	Synchronous Clock, synchronní hodinový signál
ACK	Acknowledge, potvrzení
USART	Universal Synchronous Asynchronous serial Receiver and Transmitter
PC	Personal Computer, osobní počítač
CLI	Common Intermediate Language, mezijazyk platformy .NET
CLR	Common Language Runtime, prostředí pro běh aplikací založených na platformě .NET

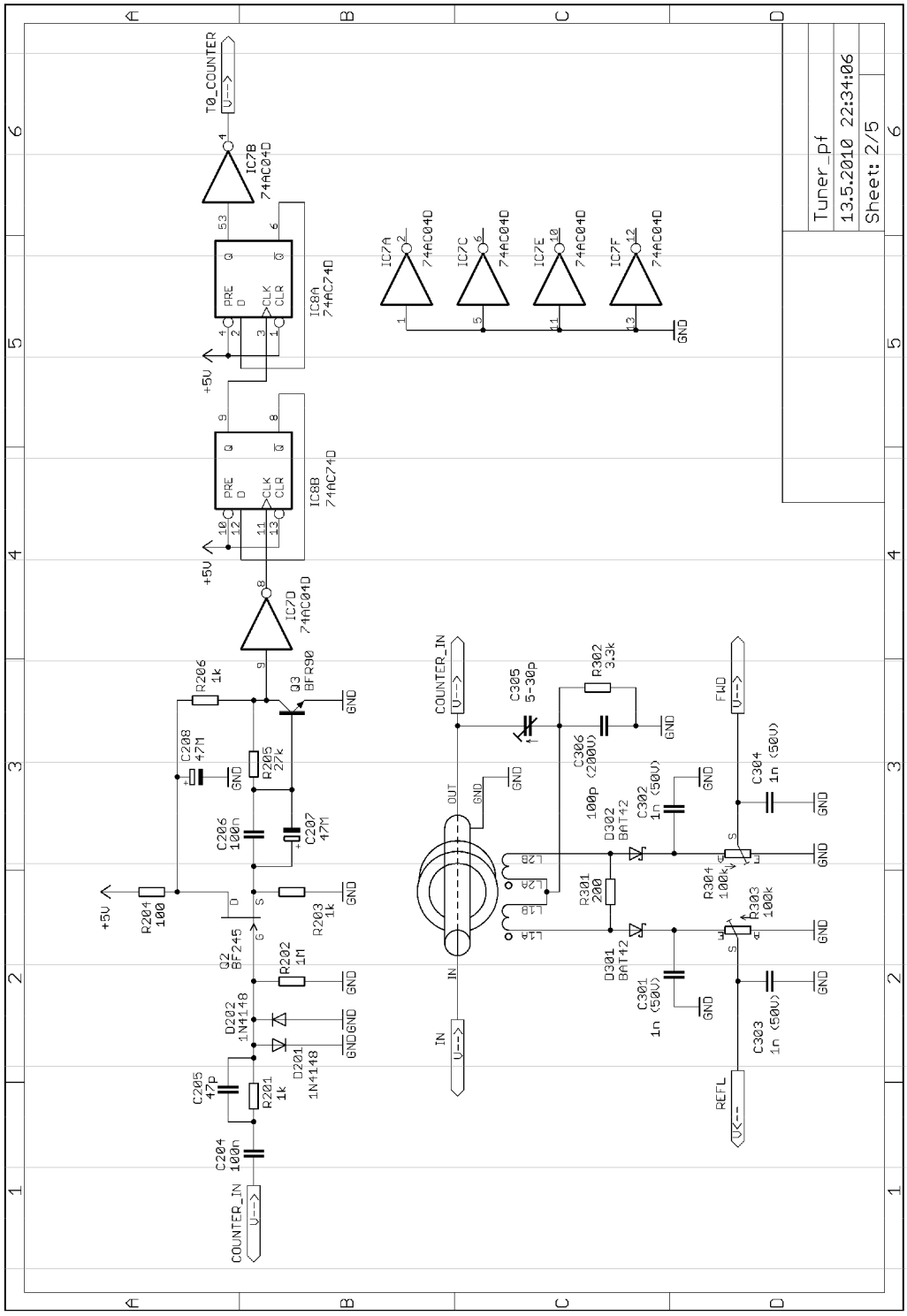
SEZNAM PŘÍLOH

A NÁVRH A REALIZACE ZAŘÍZENÍ

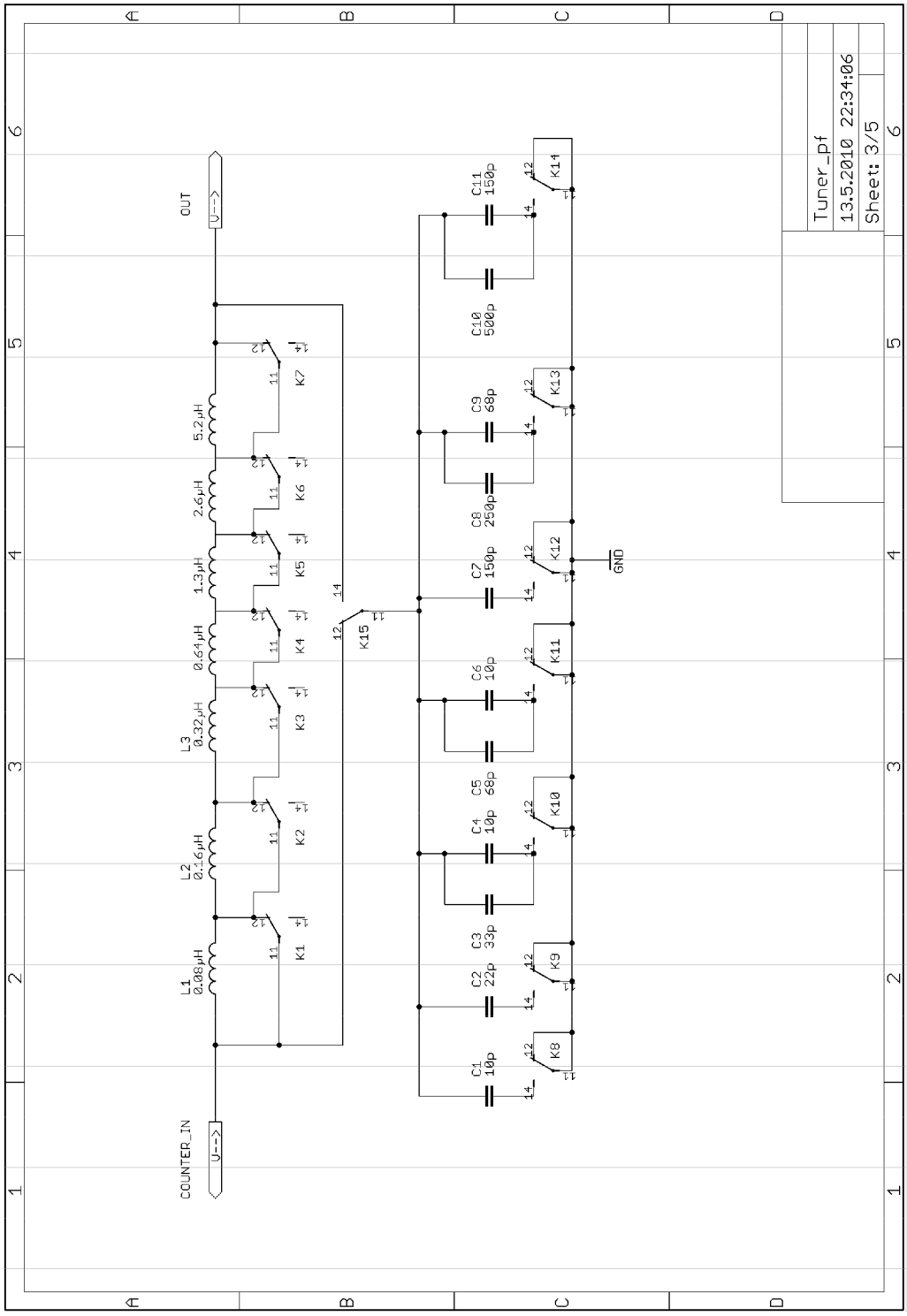
A.1 Obvodové zapojení řídící části anténního tuneru



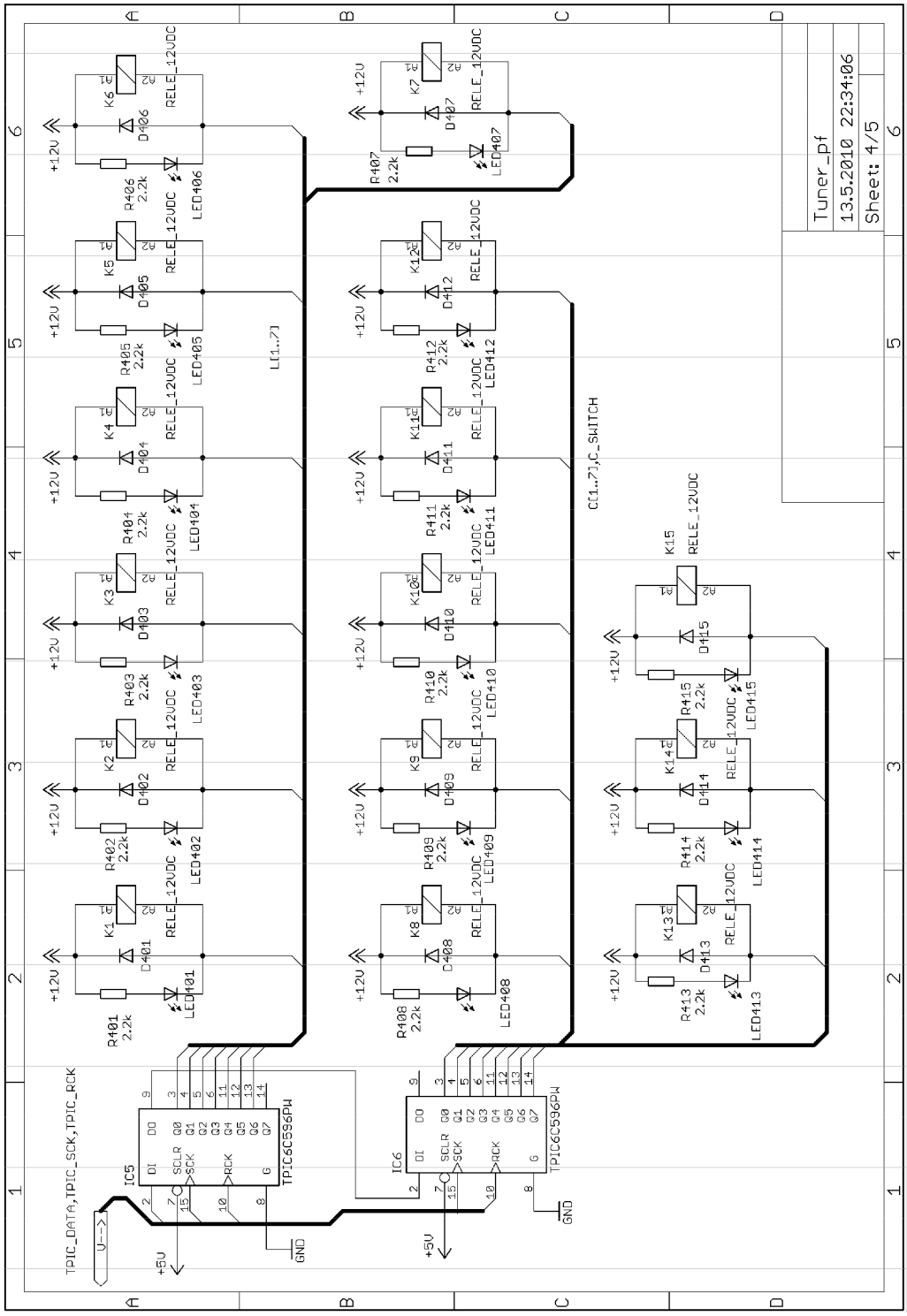
A.2 Obvodové zapojení čítače a PSV-metru



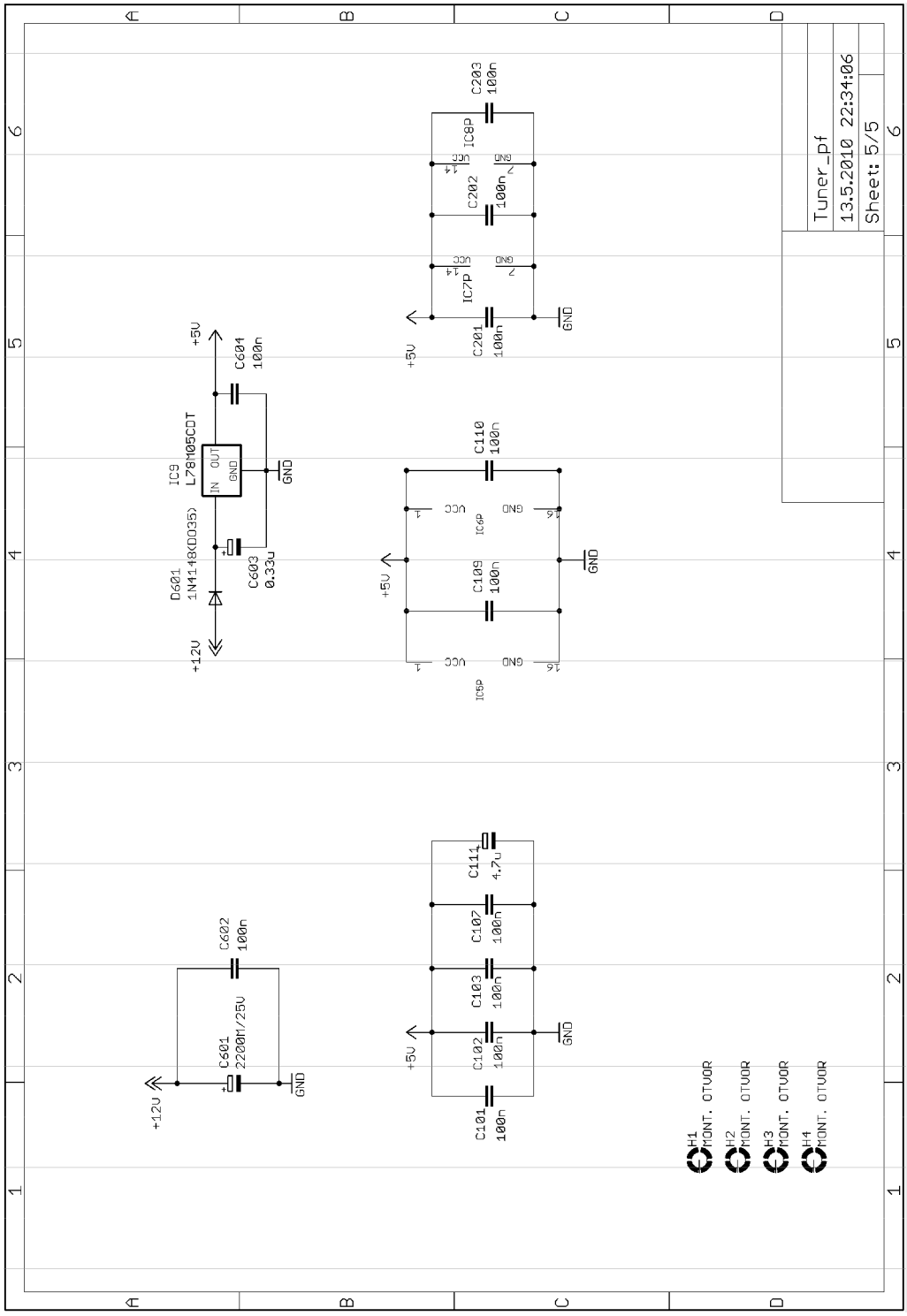
A.3 Obvodové zapojení L článku



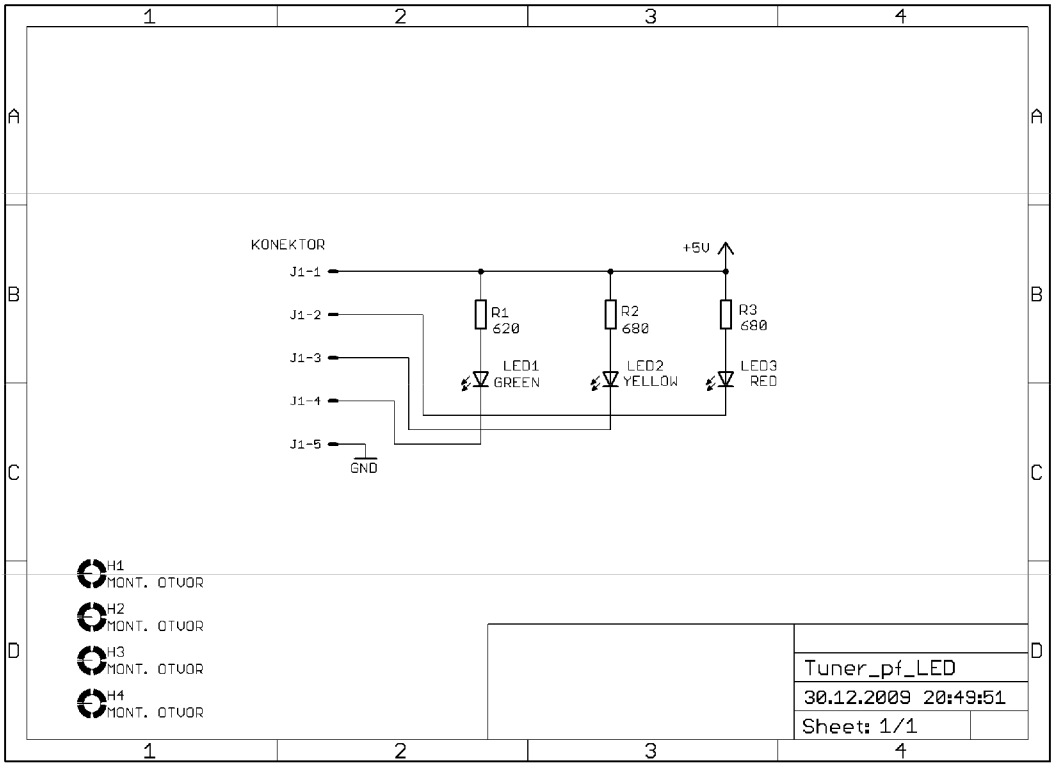
A.4 Obvodové zapojení ovládání relé



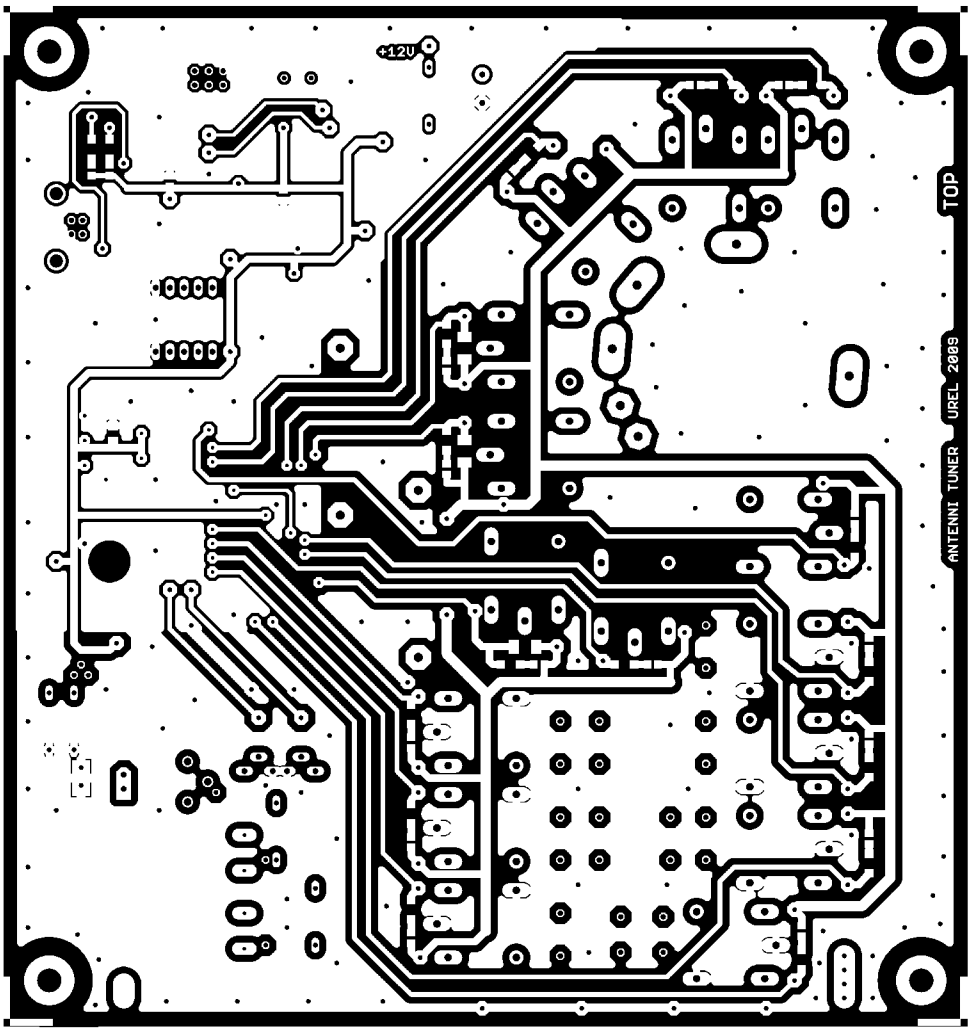
A.5 Obvodové zapojení napájení



A.6 Obvodové zapojení LED panelu

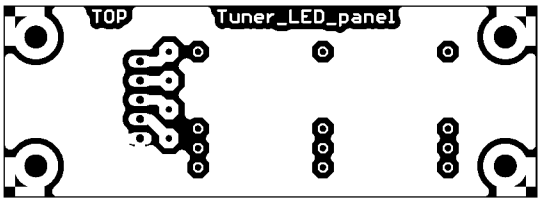


A.7 Deska plošného spoje – top (strana součástek)



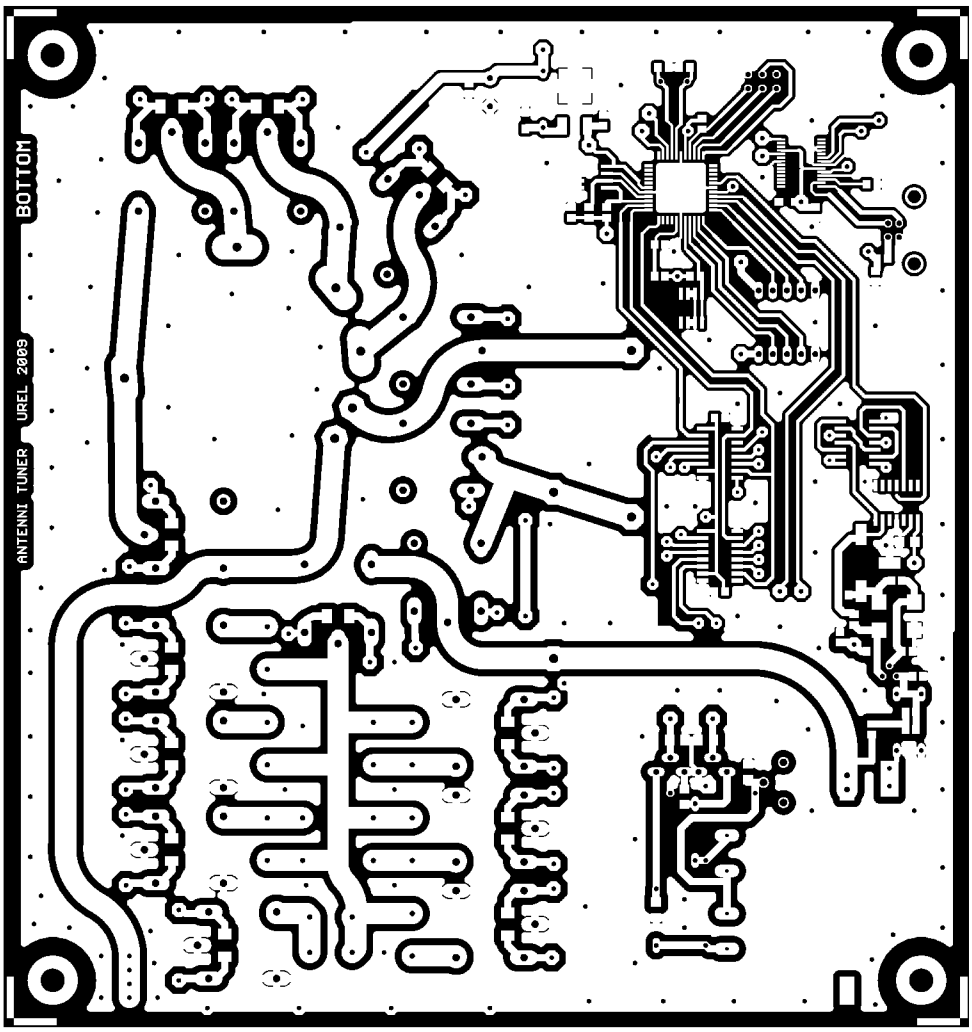
Rozměr desky 170 x 180 [mm], měřítko M3:4

A.8 Deska plošného spoje – top (LED panel)



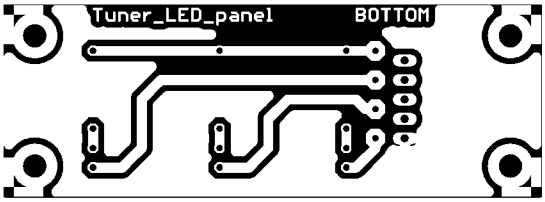
Rozměr desky 70 x 25 [mm], měřítko M1:1

A.9 Deska plošného spoje – bottom (strana spojů)



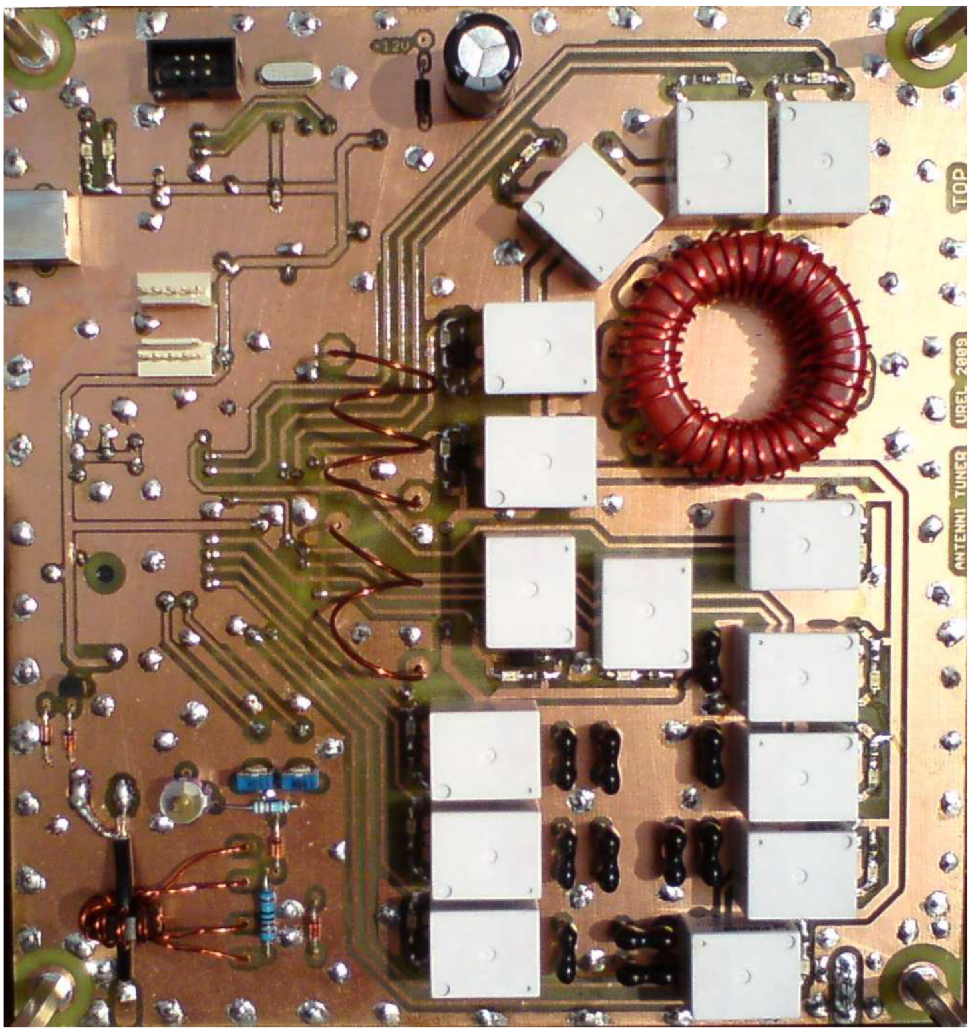
Rozměr desky 170 x 180 [mm], měřítko M3:4

A.10 Deska plošného spoje – bottom (LED panel)



Rozměr desky 70 x 25 [mm], měřítko M1:1

A.11 Osazená deska plošného spoje – top (strana součástek)

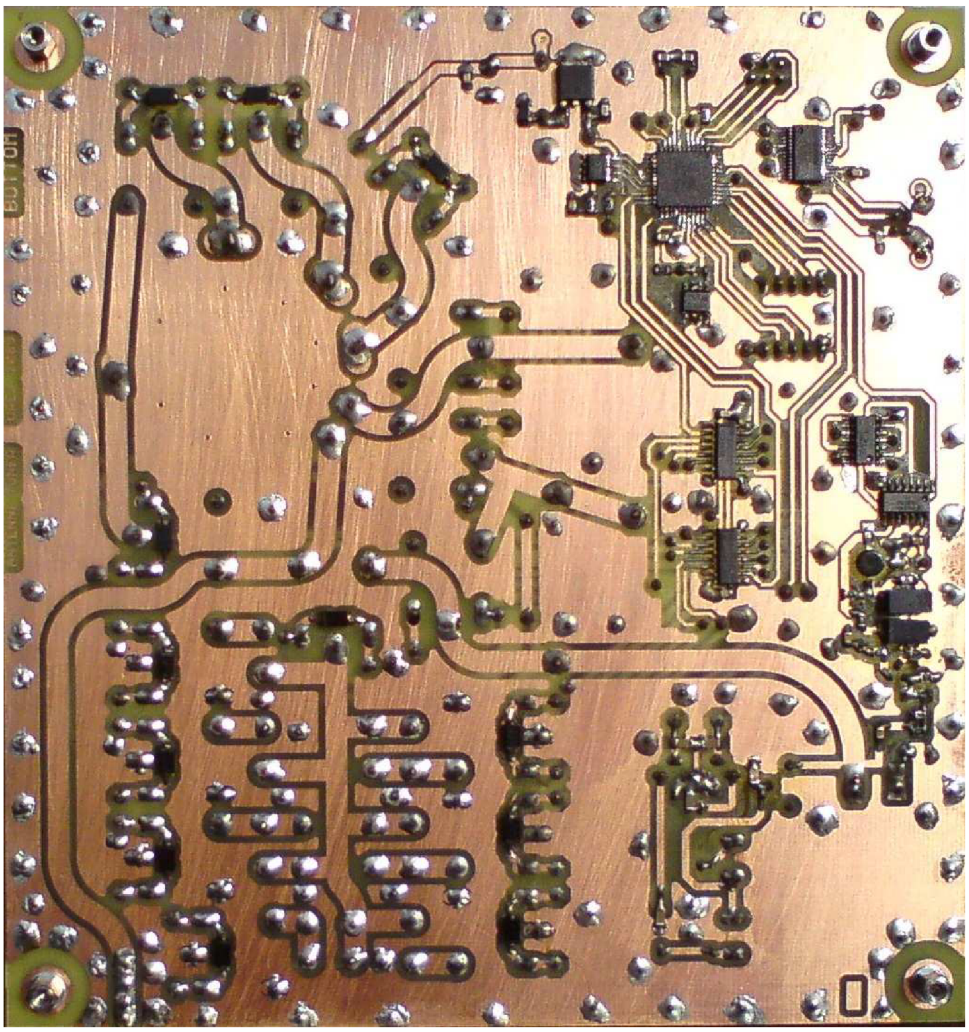


Rozměr desky 170 x 180 [mm], měřítko M3:4

A.12 Osazená deska plošného spoje – top (LED panel)

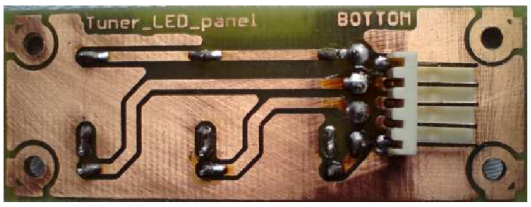
Rozměr desky 70 x 25 [mm], měřítko M1:1

A.13 Osazená deska plošného spoje – bottom (strana spojů)



Rozměr desky 170 x 180 [mm], měřítko M3:4

A.14 Osazená deska plošného spoje – bottom (LED panel)



Rozměr desky 70 x 25 [mm], měřítko M1:1

A.15 Seznam součástek

Označení	Hodnota	Pouzdro	Popis
C1, C4, C6	10 pF	C-EU075-042X103	Slídový kondenzátor
C2	22 pF	C-EU075-042X103	Slídový kondenzátor
C3	33 pF	C-EU075-042X103	Slídový kondenzátor
C5, C9	68 pF	C-EU075-042X103	Slídový kondenzátor
C7, C11	150 pF	C-EU075-042X103	Slídový kondenzátor
C8	250 pF	C-EU075-042X103	Slídový kondenzátor
C10	500 pF	C-EU075-042X103	Slídový kondenzátor
C101-C103, C106	100 nF	C0805	Keramický kondenzátor
C104, C105	22 pF	C0805	Keramický kondenzátor
C107-C110	10 nF	C0805	Keramický kondenzátor
C111	4,7 µF	SMC_A	Tantalový kondenzátor
C201-C204, C206	100 nF	C0805	Keramický kondenzátor
C205	47 pF	C0805	Keramický kondenzátor
C207, C208	47 µF	D/7343-31R	Tantalový kondenzátor
C301-C304	1 nF	C1206	Keramický kondenzátor
C305	5 - 30 pF	CTRIM808-BC	Kapacitní trimr
C306	100 pF	C1206	Keramický kondenzátor
C601	2200 µF / 25 V	E5-13	Elektrolytický kondenzátor
C602, C604	100 nF	C0805	Keramický kondenzátor
C603	0,33 µF	SMC_A	Tantalový kondenzátor
D201, D202	1N4148	DO35-10	Dioda
D301, D302	BAT42	DO35-10	Schottkyho dioda
D401-D415	1N4007	DO-214AC	Dioda
D601	1N4007	DO41-10	Dioda
IC1	ATmega16	TQFP44	Mikrokontrolér
IC2	AT24C256	SO08	Externí paměť EEPROM
IC3	TL431	SO08	Napěťová reference
IC4	FT232RL	SSOP28	Převodník UART/USB
IC5, IC6	TPIC6C596PW	SO16	Posuvný registr
IC7	74AC04D	SO14	6x invertor
IC8	74AC74D	SO14	2x klopný obvod typu D
IC9	L78M05CDT	DPACK	Napěťový regulátor 5 V / 0,5 A
J1-J3		6410-05	Konektor (5 pinů)
K1-K15	RM50	RM50	Relé 12V
L1, L2		VZD_CIVKA	Vzduchová cívka
L3	T 157-2	TOR_T157_2	Železoprachový toroid

Označení	Hodnota	Pouzdro	Popis
LED101, LED102		CHIPLED_1206	LED dioda
LED401-LED415	YELLOW	CHIPLED_1206	LED dioda
LED501	GREEN	LED5MM	LED dioda
LED502	YELLOW	LED5MM	LED dioda
LED503	RED	LED5MM	LED dioda
Q1	16 MHz	HC49/S	Krystal
Q2	BF245	TO92	Unipolární tranzistor
Q3	BFR90	SOT37	Bipolární tranzistor NPN
R101, R103	10 kΩ	R0805	Rezistor
R102, R106, R107	470 Ω	R0805	Rezistor
R104, R105	680 Ω	R0805	Rezistor
R108	4,7 kΩ	R0805	Rezistor
R109	10 kΩ	R0805	Rezistor
R201, R203, R206	1 kΩ	R0805	Rezistor
R202	1 MΩ	R0805	Rezistor
R204	100 Ω	R0805	Rezistor
R205	27 kΩ	R0805	Rezistor
R301	200 Ω (1W)	0207/15	Rezistor
R302	3,3 kΩ	0207/15	Rezistor
R303, R304	100 kΩ	CA6H	Odporový trimr
R401-R415	2,2 kΩ	R0805	Rezistor
R501	620 Ω	0207/10	Rezistor
R502, R503	680 Ω	0207/10	Rezistor
SV1		ML6	Konektor (6 pinů)
SWR1	FT 50-43	SWR_TOROID	Feritový toroid
X1		PN61729-S	Konektor USB B
X2, X3		N512	N konektor

B SOFTWARE ŘÍDÍCÍ JEDNOTKY

B.1 Zdrojový soubor Tuner_ATmega.c

```
/*
  Tuner_ATmega.c
  ----------------
  - hlavni zdrojovy soubor pro mikrokontroler ATmega16, Fcpu=16MHz
*/

//-------------------------------------------------------------------
#include "Tuner_ATmega_lib.h"
//-------------------------------------------------------------------

float napeti;
float swr;
float nove_swr;
unsigned char stav;
unsigned int Frekvence;


int main(void)
{
  led_init();
  PORTA &= ~_BV(GREEN);

  tpic_init();
  Kombinace = 0x0000;
  relay(Kombinace);

  i2c_init();
  adc_init();

  uart_init( UART_BAUD_SELECT(UART_BAUD_RATE,F_CPU) );

  // nastaveni citace/casovace 1
  // casovac 1: clock = fcpu/256, perioda preteceni cca 1s
  TCCR1B = _BV(CS12);
  TIMSK |= _BV(TOIE1);    // preruseni pri preteceni casovace 1

  stav = BEZ_SIGNALU;
  usb = ODPOJENO;

  sei();

  while(1)
  {
        if ((stav == POTREBA_LADIT) & (usb == ODPOJENO))
        {
              PORTA &= ~_BV(YELLOW);

              TCCR1B = 0x00;          // vypnuti casovace 1
              TIMSK &= ~_BV(TOIE1);
              TIFR |= _BV(TOV1);

              Frekvence = getFrequency(1);
```

```
                if ((Frekvence < 31000) & (Frekvence > 1000))
                // rozmezi pasma pro ladeni [kHz]
                {
                      PORTA &= ~_BV(RED);

                      Kombinace = load_comb(Frekvence);
                      relay(Kombinace);

                      swr = getSWR();
                      napeti = getVoltage(CHANNEL_1);

                      if((swr > 1.5) & (napeti > VOLTAGE_THRESHOLD))
                            {
                                  Kombinace = Tuning();
                                  save_comb(Frekvence, Kombinace);
                                  swr = getSWR();
                            }
                            PORTA |= _BV(RED);
                      }
                      stav = NALADENO;

                      TCCR1B = _BV(CS12);     // zapnuti casovace 1
                      TIMSK |= _BV(TOIE1);

                      PORTA |= _BV(YELLOW);
                }
          RemoteControl();
    }
    return 0;
}

ISR( TIMER1_OVF_vect )    // obsluha preteceni casovace 1
{
    PORTA &= ~_BV(YELLOW);
    _delay_ms(50);

    if (stav == BEZ_SIGNALU)
    {
          napeti = getVoltage(CHANNEL_1);

          if(napeti > VOLTAGE_THRESHOLD)      // napeti alespon 100mV
          {
                stav = POTREBA_LADIT;
          }
    }
    if (stav == NALADENO)   // opetovne ladeni, kdyz se SWR zvetsi
    {
          nove_swr = getSWR();
          if (nove_swr > (swr + 0.8)) stav = POTREBA_LADIT;
    }
    PORTA |= _BV(YELLOW);
}
```

B.2 Hlavičkový soubor Tuner_ATmega_lib.h

```
/*
  Tuner_ATmega_lib.h
  ------------------
        - hlavickovy soubor knihovny funkci
*/

//-------------------------------------------------------------------
#ifndef TUNER_ATMEGA_LIBRARY
#define TUNER_ATMEGA_LIBRARY
//-------------------------------------------------------------------

#define DELAY_BEFORE_SWITCH     5     // zpozdeni pri spinani rele
#define DELAY_AFTER_SWITCH      50
// prahove napeti [V] pro indikaci pritomnosti signalu
#define VOLTAGE_THRESHOLD       0.1

#define GREEN       2
#define YELLOW      3     // indikacni panel s diodami (PORT A)
#define RED         4

#define TPIC_RCK    4     // piny pro komunikaci s posuvnymi
#define TPIC_SCK    3     // registry TPIC (PORT C)
#define TPIC_DATA   2

#define CHANNEL_0   0     // kanaly A/D prevodniku
#define CHANNEL_1   1

#define BEZ_SIGNALU       0
#define POTREBA_LADIT     1     // stav tuneru
#define NALADENO          2

#define ODPOJENO    0           // stav vzdaleneho rizeni pres USB
#define PRIPOJENO   1

#define LNetwork    0     // L clanek
#define GamaNetwork 1     // Gama clanek

#define ADRESA_EEPROM     0xA0  // I2C adresa externi pameti EEPROM

#define UART_BAUD_RATE    9600  // UART symbolova rychlost: 9600 Bd

#ifndef F_CPU                   // kmitocet CPU [Hz]
#define F_CPU 16000000UL
#endif
//-------------------------------------------------------------------
#include <avr/io.h>       // knihovna popisujici mikrokontroler
#include <avr/interrupt.h>      // knihovna pro preruseni
#include <util/delay_basic.h>   // knihovna pro kratka zpozdeni
#include <util/delay.h>   // knihovna pro zpozdeni v [ms] pro rele

#include <tuner/i2cmaster.h>    // knihovna pro komunikaci pres I2C
#include <tuner/uart.h>   // knihovna pro komunikaci pres UART
#include <avr/pgmspace.h>

#include <stdlib.h>       // pouziti funkce dtostre
//-------------------------------------------------------------------
```

```
unsigned int Kombinace;         // globalni promenne
unsigned char usb;
//-------------------------------------------------------------------
void led_init(void);
void tpic_init(void);
void relay (unsigned int combination);
unsigned int create_comb(unsigned char coil, unsigned char
    capacitor, unsigned char network);
unsigned char swap_7_bits(unsigned char byte);
unsigned char save_comb(unsigned int freq_kHz, unsigned int
    combination);
unsigned int load_comb(unsigned int freq_kHz);
unsigned long int getFrequency(char unit);
void adc_init (void);
float getVoltage (unsigned char channel);
float getSWR (void);
unsigned char UartError (unsigned int znak);
unsigned char RemoteControl(void);
unsigned int Tuning (void);
//-------------------------------------------------------------------
#endif
```

B.3 Knihovna funkcí Tuner_ATmega_lib.c

```
/*
  Tuner_ATmega_lib.c
  ------------------
        - knihovna funkci
*/
//-------------------------------------------------------------------
#include "Tuner_ATmega_lib.h"
//-------------------------------------------------------------------

// globalni promenne
volatile unsigned long int counter0_overflows;
// pouzit dat. typ long int, protoze hodnota nabyva az 160 000

volatile unsigned int timer2_overflows;
volatile unsigned long int counted_periods;
//-------------------------------------------------------------------
// inicializace komunikace s LED panelem, volana jen jednou

void led_init(void)
{
  // nastaveni vystupnich pinu
  DDRA |= _BV(GREEN)|_BV(YELLOW)|_BV(RED);
  //log.1 na pinech, vsechny diody zhasnute
  PORTA |= _BV(GREEN)| _BV(YELLOW)| _BV(RED);
}
//-------------------------------------------------------------------
// inicializace komunikace s obvody TPIC, volana jen jednou

void tpic_init(void)
{
  // nastaveni vystupnich pinu pro komunikaci s TPIC
  DDRC = _BV(TPIC_RCK)|_BV(TPIC_SCK)|_BV(TPIC_DATA);
  //log. 0 na vsechny piny
```

```
  PORTC = ~_BV(TPIC_RCK) &~_BV(TPIC_SCK)& ~_BV(TPIC_DATA);
}
//---------------------------------------------------------------------
// ovladani rele pres obvody TPIC

void relay (unsigned int combination)
{
  char count;

  for(count=0;count<16;count++)
  {     // maskovani konkretniho bitu (rele) z kombinace
        if (combination & _BV(count))
              PORTC |= _BV(TPIC_DATA);  // log. 1 na vystup DATA

        else  PORTC &= ~_BV(TPIC_DATA); // log. 0 na vystup DATA

        //kladny impulz SCK, posun dat v registru
        PORTC |= _BV(TPIC_SCK);
        _delay_loop_1(5);
        PORTC &= ~_BV(TPIC_SCK);
  }
  _delay_loop_1(10);
  //kladny impulz RCK, vystaveni dat v registru na vystup
  PORTC |= _BV(TPIC_RCK);
  _delay_loop_1(5);
  PORTC &= ~_BV(TPIC_RCK);
}
//---------------------------------------------------------------------
// vytvoreni kombinace LC (2 byty) z hodnot L, C a topologie site

unsigned int create_comb(unsigned char coil, unsigned char
  capacitor, unsigned char network)
{
  return
  (swap_7_bits(coil)<<9)+(swap_7_bits(capacitor)<<1)+network;
}
//---------------------------------------------------------------------
// prerovnani 7 bitu v poradi MSB-LSB na LSB-MSB

unsigned char swap_7_bits(unsigned char byte)
{
  unsigned char count;
  unsigned char swapped=0;

  for (count=0;count<7;count++)
        swapped += ((byte & _BV(6-count))>0)<<count;
  return swapped;
}
//---------------------------------------------------------------------
// ulozeni kombinace LC (2 byty) do pameti EEPROM na pozici danou
// kmitoctem signalu, f= 0-32000 kHz, krok 4 kHz, 8000 pozic

unsigned char save_comb(unsigned int freq_kHz,unsigned int
  combination)
{
  unsigned int addr;
  // deleni ctyrmi se zaokrouhlenim dolu, vytvoreni adres
  addr = freq_kHz/4;
  addr = 2*addr;    // jeden zaznam zabira 2 byty pameti
```

```
  // 1. byte
  //odeslani adresy I2C zarizeni, rezim zapisu
  i2c_start_wait(ADRESA_EEPROM+I2C_WRITE);
  //zapis horniho bytu adresy pametove bunky
  // + kontrola, zda I2C zarizeni odpovida
  if(i2c_write(addr>>8)) return 1;
  //zapis dolniho bytu adresy pametove bunky
  i2c_write(addr);
  //zapis hodnoty do pameti - nizsi byte kombinace
  i2c_write(combination);
  i2c_stop();             //ukoncovaci podminka

  // 2. byte
  addr++;
  i2c_start_wait(ADRESA_EEPROM+I2C_WRITE);
  if(i2c_write(addr>>8)) return 1;
  i2c_write(addr);
  //zapis hodnoty do pameti - vyssi byte kombinace
  i2c_write(combination>>8);
  i2c_stop();

  return 0;
}
//------------------------------------------------------------------
// nacteni kombinace LC (2 byty) z pameti EEPROM

unsigned int load_comb(unsigned int freq_kHz)
{
  unsigned int addr;
  unsigned int data;

  addr = freq_kHz/4;
  addr = 2*addr;

  i2c_start_wait(ADRESA_EEPROM+I2C_WRITE);
  i2c_write(addr>>8); // zapsani adresy pametove bunky pro cteni
  i2c_write(addr);

  i2c_rep_start(ADRESA_EEPROM+I2C_READ);
  // cteni 1. bytu z pameti (nizsi), pozadavek na cteni dalsiho
  data = i2c_readAck();
  // cteni 2. bytu z pameti (vyssi), neni pozadovan dalsi
  data = (i2c_readNak()<<8) + data;
  i2c_stop();

  return data;
}
//------------------------------------------------------------------
// mereni kmitoctu

ISR( TIMER0_OVF_vect )    // obsluha preteceni citace 0
{
  counter0_overflows++;
}

ISR( TIMER2_OVF_vect )    // obsluha preteceni casovace 2
{
  counted_periods += (counter0_overflows<<8) + TCNT0;
```

```
  TCNT0 = 0;                    // vynulovani citace 0
  counter0_overflows = 0;

  timer2_overflows++;
}

unsigned long int getFrequency(char unit)
// 0 - kmitocet [Hz], 1 - kmitocet [kHz]
{
  float average_periods;
  unsigned long frequency;

  counter0_overflows = 0;
  timer2_overflows = 0;
  TCNT0 = 0;              // vynulovani citacu/casovacu
  TCNT2 = 0;
  counted_periods = 0;

  // nastaveni citacu/casovacu
  // citac 0: externi hodinovy vstup T0, vzestupna hrana
  TCCR0 = _BV(CS02) | _BV(CS01) | _BV(CS00);
  // casovac 2: clock = fcpu/1024
  TCCR2 = _BV(CS22) | _BV(CS21) | _BV(CS20);
  // povoleni preruseni
  TIMSK |= _BV(TOIE2) | _BV(TOIE0);

  while(timer2_overflows<50);   // ulozeni 50 hodnot

  TIMSK = 0x00;     // zakazani preruseni
  TCCR0 = 0x00;     // zastaveni citacu/casovacu
  TCCR2 = 0x00;
  // vypocet zmereneho kmitoctu, prumerovani 50 hodnot
  average_periods = ((float) counted_periods) / 50.000;
  frequency = (unsigned long) 4* (average_periods * 61.0534);

  if(unit == 0) return frequency;     // kmitocet v Hz
  else return frequency/1000;         // kmitocet v kHz
}
//---------------------------------------------------------------------
// inicializace A/D prevodniku, volana jen jednou

void adc_init (void)
{
  DDRA &= ~_BV(CHANNEL_0)& ~_BV(CHANNEL_1); // piny jako vstupni
  // nastaveni externi ref. hodnoty (2,5V) a kanalu 1 pro prevod
  ADMUX = _BV(MUX0);
  ADCSRA   =   _BV(ADEN)|_BV(ADSC)|_BV(ADPS2)|_BV(ADPS1)|_BV(ADPS0);

  // ADEN - zapnuti A/D prevodniku
  // ADSC - zahajeni prvniho prevodu
  // ADPSn - delicka hodinoveho signalu f/128

  // cekani na dokonceni A/D prevodu
  while(bit_is_clear(ADCSRA,ADIF));
  // po provedeni prvniho prevodu vynulovani bitu ADIF (zapis 1)
  ADCSRA |= _BV(ADIF);
}
//---------------------------------------------------------------------
// napeti na zvolenem kanalu A/D prevodniku, prumer 5 hodnot
```

```
float getVoltage (unsigned char channel) // vyber kanalu 0 nebo 1
{
  unsigned int voltage_sum = 0;
  float voltage;
  char count;

  if(channel != 0)  ADMUX |= _BV(MUX0);     // vyber kanalu 1 (FWD)
  else              ADMUX &= ~_BV(MUX0);    // vyber kanalu 0 (REF)

  for(count=0; count<5; count++)
  {
        ADCSRA |= _BV(ADSC);    // zahajeni A/D prevodu
        while(bit_is_clear(ADCSRA,ADIF));
        ADCSRA |= _BV(ADIF);
        voltage_sum += ADCW;
  }
  // vypocet napeti, prumerovani 5 hodnot
  voltage = ((((float)voltage_sum) * 2.5) / 5115.000);
  return voltage;
}
//---------------------------------------------------------------------
// vypocte hodnotu PSV z dat poskytnutych A/D prevodnikem

float getSWR (void)
{
  unsigned int fwd_sum = 0;     // suma zmerenych napeti FWD
  unsigned int ref_sum = 0;     // suma zmerenych napeti REF
  float fwd_voltage, ref_voltage, swr, divider;
  char count;

  for(count=0; count<20; count++)
  {
        ADMUX &= ~_BV(MUX0);    // vyber kanalu 0 (REF)
        ADCSRA |= _BV(ADSC);    // zahajeni A/D prevodu

        while(bit_is_clear(ADCSRA,ADIF));
        ADCSRA |= _BV(ADIF);
        ref_sum += ADCW;  // pricteni hodnoty A/D prevodu k sume

        ADMUX |= _BV(MUX0);     // vyber kanalu 1 (FWD)
        ADCSRA |= _BV(ADSC);    // zahajeni A/D prevodu

        while(bit_is_clear(ADCSRA,ADIF));
        ADCSRA |= _BV(ADIF);
        fwd_sum += ADCW;
  }
  // prumerovani 20 hodnot (20*1023) a prepocet na U, Uref = 2,5V

  fwd_voltage = (( ((float) fwd_sum )* 2.5) / 20460.000000);
  ref_voltage = (( ((float) ref_sum )* 2.5) / 20460.000000);

  // vypocet pomeru stojatych vln
  divider = fwd_voltage - ref_voltage;
  if (divider > 0)  // osetreni, aby nedoslo k deleni nulou
        swr = (fwd_voltage + ref_voltage) / divider;
  else swr = 999.0;

  if (swr > 999.0) return 999.0;      // uprava velkych hodnot swr
  else  return swr;
```

```
}
//---------------------------------------------------------------------
// inteligentni algoritmus ladeni, vraci hodnotu vysledne kombinace

unsigned int Tuning (void)
{
// indexy reperezentuji aktualne vybranou hodnotu parametru L a C
  unsigned char L_index, C_index;
  float SWR;        // aktualni pomer stojatych vln
  unsigned char L_best, C_best; // indexy kombinace s nejlepsim PSV
  float SWR_best;         // nejlepsi dosazeny pomer stojatych vln
  // sedmibitove kombinace reprezentujici vybrane prvky L nebo C
  unsigned char coil, capacitor;
  // pro urceni, jaky clanek byl vybrana, 0=L clanek 1=gama clanek
  unsigned char network;

  // 1. etapa - vyber clanku a nejlepsi z 8x8 kombinaci
  //-------------------------------------------------
  network = GamaNetwork;
  L_best = 0;
  C_best = 0;
  // pokud nebude mereni PSV dobre fungovat (napr. neni signal),
  // bude vybrana nejmensi kombinace = nebudou sepnuty zadne prvky
  SWR_best = 999.0;

  // Gama clanek
  // indexy nabyvaji hodnot 1 az 8 v sestupnem poradi
  // pri vice kombinacich se stejne dobrym PSV, bude vybrana ta
  // s mensi hodnotou parametru

  for(C_index = 8; C_index>0; C_index--)
   for(L_index = 8; L_index>0; L_index--)
   {
        _delay_ms(DELAY_BEFORE_SWITCH);
        // vytvoreni kombinace a sepnuti rele
        relay(create_comb((L_index-1)<<4,           (C_index-1)<<4,
          GamaNetwork));
        _delay_ms(DELAY_AFTER_SWITCH);

        SWR = getSWR();   // mereni pomeru stojatych vln

        if(SWR<=SWR_best) // porovnani s nejlepsim PSV
        {
              SWR_best = SWR;
              L_best = L_index-1;
                    C_best = C_index-1;
        }
   }
  // L clanek
  for(C_index = 8; C_index>0; C_index--)
   for(L_index = 8; L_index>0; L_index--)
   {
        _delay_ms(DELAY_BEFORE_SWITCH);
        relay(create_comb((L_index-1)<<4,           (C_index-1)<<4,
          LNetwork));
        _delay_ms(DELAY_AFTER_SWITCH);
        SWR = getSWR();

        if(SWR<=SWR_best)
```

```
      {// PSV kombinace s L clankem je lepsi, jak s gama clankem
            network = LNetwork;
            SWR_best = SWR;
            L_best = L_index-1;
            C_best = C_index-1;
            }
      }
coil = (L_best & 0x06)<<5;// ulozeni dvou nejvetsich prvku L a C
capacitor = (C_best & 0x06)<<5;

// 2. etapa - vyber z hodnot 8x8 okolo vybr. bodu v rovine LC
//-----------------------------------------------------------------
L_best = 0;
C_best = 0;
SWR_best = 999.0;

for(C_index = 8; C_index>0; C_index--)
 for(L_index = 8; L_index>0; L_index--)
 {
      _delay_ms(DELAY_BEFORE_SWITCH);
      relay(create_comb(coil   |   (L_index-1)<<2,   capacitor   |
       (C_index-1)<<2, network));
      _delay_ms(DELAY_AFTER_SWITCH);
      SWR = getSWR();   // mereni pomeru stojatych vln

      if(SWR<=SWR_best) // porovnani s nejlepsim PSV
      { // nahrada predchozi kombinace s nejlepsim PSV
            SWR_best = SWR;
            L_best = L_index-1;
                  C_best = C_index-1;
      }
 }    // ulozeni 3. a 4. nejvetsiho prvku L a C
coil = coil | (L_best & 0x06)<<3;
capacitor = capacitor | (C_best & 0x06)<<3;

// 3. etapa
//---------
L_best = 0;
C_best = 0;
SWR_best = 999.0;

for(C_index = 8; C_index>0; C_index--)
 for(L_index = 8; L_index>0; L_index--)
 {
      _delay_ms(DELAY_BEFORE_SWITCH);
      relay(create_comb(coil | (L_index-1), capacitor | (C_index-
        1), network));
      _delay_ms(DELAY_AFTER_SWITCH);
      SWR = getSWR();

      if(SWR<=SWR_best)
      {
            SWR_best = SWR;
            L_best = L_index-1;
            C_best = C_index-1;
      }
 }
coil = coil | L_best;   // ulozeni tri nejmensich prvku L a C
capacitor = capacitor | C_best;
```

```
  // sepnuti vybrane nejlepsi kombinace
  relay(create_comb(coil, capacitor, network));
  // funkce vraci hodnotu kombinace LC
  return create_comb(coil, capacitor, network);
}
//------------------------------------------------------------------
// vyhodnoceni chyby prijateho znaku (UART) a odeslani kodu chyby

unsigned char UartError (unsigned int znak)
{
  char chyba = 0;

  if (znak & UART_FRAME_ERROR) chyba = 1; // UART Frame Error
  if (znak & UART_OVERRUN_ERROR) chyba = 2; // UART Overrun Error
  if (znak &UART_BUFFER_OVERFLOW)chyba=3; // Buffer overflow Error

  if (chyba != 0)
  {
        uart_putc (1);          // posle se prvni znak 1
        uart_putc (chyba);      // a druhy obsahuje kod chyby
              return 1;
  }
  else return 0;
}
//------------------------------------------------------------------
// rizeni veskere komunikace s PC

unsigned char RemoteControl(void)
{
  unsigned int znak;      // prijaty znak
  char prichoziData[3];   // 3 prijate znaky
  char odchoziData[12];   // az 12 odchozich znaku
  char index;             // poradi prijateho znaku
  char error = 0;
  float SWR, napeti;      // merene hodnoty
  unsigned long int kmitocet;

  znak = uart_getc();     // nacteni znaku ze zasobniku prijimace
  // jsou v zasobniku nejaka data?
  if (!(znak & UART_NO_DATA) )
  {
   if(!UartError(znak))   // osetreni chyby pri prijmu
   {
    switch (znak)         // reakce na prijaty prikaz
    {
     case 1:  // pripojeni vzdaleneho rizeni
     {
        TCCR1B = 0x00;    // vypnuti casovace 1
        TIMSK &= ~_BV(TOIE1);
        TIFR |= _BV(TOV1);

        usb = PRIPOJENO;
        // zluta LED indikuje vzdalene rizeni
        PORTA &= ~_BV(YELLOW);

        uart_putc(znak);
     } break;

     case 2:  // odpojeni vzdaleneho rizeni
```

```
{
   usb = ODPOJENO;
   PORTA |= _BV(YELLOW);
   uart_putc(znak);
   TCCR1B = _BV(CS12);     // zapnuti casovace 1
   TIMSK |= _BV(TOIE1);
} break;

case 10: // zaslat hodnotu Kombinace LC
{
   uart_putc( (unsigned char)znak );
   uart_putc( Kombinace>>8 );
   uart_putc( Kombinace );
} break;

case 20: // sepnout zasilanou Kombinaci LC
{
   prichoziData[0]   = (unsigned char) znak;
   for(index=0;index<2;index++)
   {     // cekani na prijem dalsich znaku
    do   znak = uart_getc();
    while (znak & UART_NO_DATA);

    if(UartError(znak)) // osetreni chyb prijimanych znaku
    {
         error = 1;
         break;
    }
    prichoziData[index+1] = (unsigned char) znak;
   }
   if(!error)
   {
         uart_putc(prichoziData[0]);
         uart_putc(prichoziData[1]);
         uart_putc(prichoziData[2]);

         Kombinace = (prichoziData[1]<<8) + prichoziData[2];
         relay(Kombinace);
   }
} break;

case 30: // zaslat napeti na kanalu 1 A/D prevodniku
{
   napeti = getVoltage(1);
   // textove vyjadreni zacina znamenkem, 11 znaku + znak NULL
   dtostre((double) napeti, odchoziData, 4, 0x02);

   uart_putc(znak);
   uart_puts(odchoziData);
} break;

case 31: // zaslat napeti na kanalu 0 A/D prevodniku
{
   napeti = getVoltage(0);
   dtostre((double) napeti, odchoziData, 4, 0x02);

   uart_putc(znak);
   uart_puts(odchoziData);
} break;
```

```
      case 32: // zaslat hodnotu pomeru stojatych vln
      {
         SWR = getSWR();
         dtostre((double) SWR, odchoziData, 4, 0x02);

         uart_putc(znak);
         uart_puts(odchoziData);
      } break;

      case 33: // zap/vypnout pull-up rezistory na u A/D prevodniku
      {
         if(bit_is_clear(PORTA,0))
               PORTA |= _BV(CHANNEL_0) | _BV(CHANNEL_1);
         else PORTA &= ~_BV(CHANNEL_0) & ~_BV(CHANNEL_1);

         uart_putc(znak);
      } break;

      case 40: // zaslat hodnotu kmitoctu
      {
         PORTA &= ~_BV(RED);     // indikace cervenou LED
         kmitocet = getFrequency(1);

         uart_putc( (unsigned char)znak );
         uart_putc( kmitocet>>24 );
         uart_putc( kmitocet>>16 );
         uart_putc( kmitocet>>8 );
         uart_putc( kmitocet );

         PORTA |= _BV(RED);
      } break;

      case 50: // spustit algoritmus automatickeho ladeni
      {
         PORTA &= ~_BV(RED);     // indikace cervenou LED

         uart_putc(znak);
         Kombinace = Tuning();

         PORTA |= _BV(RED);
      } break;

      case 60: // ulozit Kombinaci LC do pameti EEPROM
      {
         PORTA &= ~_BV(RED);

         kmitocet = getFrequency(1);
         save_comb(kmitocet, Kombinace);

         uart_putc(znak);

         PORTA |= _BV(RED);
      } break;

      default: uart_putc( (unsigned char)znak );
     }
    }
   } return 0;
}
```