Fakulta elektrotechnická
Katedra aplikované elektroniky a telekomunikací

Bakalářská práce

Dálkové ovládání pro vjezdové brány

Autor práce: Petr Gallistl
Vedoucí práce: Ing. Kamil Kosturik, Ph.D.

Plzeň 2013

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta elektrotechnická
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Petr GALLISTL**
Osobní číslo:	**E10B0297P**
Studijní program:	**B2612 Elektrotechnika a informatika**
Studijní obor:	**Elektronika a telekomunikace**
Název tématu:	**Dálkové ovládání pro vjezdové brány**
Zadávající katedra:	**Katedra aplikované elektroniky a telekomunikací**

Zásady pro vypracování:

1. Navrhněte vhodnou technologii pro bezdrátové ovládání brány.
2. Navrhněte jednotku přijímače a vysílače bezdrátového ovládání.
3. Navržené řešení realizujte.
4. Zhodnoťte navržené zařízení, diskutujte možnosti případného vylepšení.

Rozsah grafických prací: **podle doporučení vedoucího**

Rozsah pracovní zprávy: **20 - 30 stran**

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.

Vedoucí bakalářské práce: **Ing. Kamil Kosturik, Ph.D.**
Katedra aplikované elektroniky a telekomunikací

Konzultant bakalářské práce: **Ing. Kamil Kosturik, Ph.D.**
Katedra aplikované elektroniky a telekomunikací

Datum zadání bakalářské práce: **15. října 2012**

Termín odevzdání bakalářské práce: **7. června 2013**

L.S.

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

Doc. Dr. Ing. Vjačeslav Georgiev
vedoucí katedry

V Plzni dne 15. října 2012

Abstrakt

V této práci budou řešeny jak teoretické znalosti potřebné k navržení dálkového ovládání, tak i znalosti potřebné k jeho realizaci. V teoretické části bude řešena použitá modulace On-Off keying, která je podskupinou modulace ASK. Také zde budou řešeny moduly přijímače tak i vysílače. Na závěr teoretické části se objeví základní popis mikroprocesoru AVR ATmega8. V praktické části bude popsaná realizace desky plošných spojů v programu EAGLE a také zde bude popis použitého programu...

Klíčová slova

Dálkové ovládání, Rádiové ovládání, AVR, ATmega8, USART, OOK

Abstract

Gallistl, Petr. *Remote control for entrance gates* [*Dálkové ovládání pro vjezdové brány*]. Pilsen, 2013. Bachelor thesis (in Czech). University of West Bohemia. Faculty of Electrical Engineering. Department of Applied Electronics and Telecommunications. Supervisor: Kamil Kosturik

---

The present thesis is focused on remote control for entrance gate. In this thesis will be presented theoretical knowledge about OOK modulation, which is subset of ASK modulation. Will be also describing modules of receivers and transmitters, also will be described mikroprocessor AVR ATmega8. The next part will be about realization PCB in CAD system EAGLE and it will be shown code. ...

Keywords

Remote control, Radio control, AVR, ATmega8, USART, OOK

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci, zpracovanou na závěr studia na Fakultě elektrotechnické Západočeské univerzity v Plzni.

Prohlašuji, že jsem svou závěrečnou práci 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 závěrečné 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 § 270 trestního zákona č. 40/2009 Sb.

Také prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

V Plzni dne 6. června 2013

Petr Gallistl

.....

Podpis

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů a zkratek

ASK	Amplitude swift keying.
OOK	On-Off keying.
f	Frekvence $[Hz]$.
V_m	Modulační rychlost $[Baud]$.
V_p	Přenosová rychlost $\left[\frac{bit}{s}\right]$.
GPR	General purpose registers. Registry pro všeobecné použití.
PWM	Pulse wide modulation. Pulsně šířková modulace.
SPI	Serial Peripheral Interface. Sériové rozhraní
TWI	Two wire interface. Dvou vodičové rozhraní.
DPS	Deska plošných spojů.
VCC	Napájecí napějí.
GND	Zemnění.
SMT	Surface mount technology. Technologie pro povrchovou montáž.
SMD	Surface mount device. Zařízení pro povrchovou montáž.
vf	Vysokofrekvenční.
nf	Nízkofrekvenční.
BOD	Brown Out Detection. Detekce podpětí
JTAG	Joint Test Action Group. Testovací rozhraní pro procesor.

1

Úvod

Tato bakalářská práce je zaměřena na návrh a realizaci bezdrátového ovládání pro vjezdovou bránu. Tímto problémem se zabývá mnoho specializovaných firem, které řeší celou problematiku od vysílacích a přijímacích modulů, až po samotnou realizaci celé brány. Nicméně toto řešení bývá relativně nákladné vzhledem k použitým technologiím a proto jsem se rozhodl, že se pokusím podobné zařízení realizovat v rámci mé bakalářské práce.

Tato práce je rozdělena do čtyřech kapitol. První kapitola se zabývá teoretickým rozborem problému jako je legislativa pro vysílání v České republice. Také se v této kapitole řeší použitá technologie pro vysílání a přijímání signálu. Další kapitola se zabývá mikroprocesorem Atmel z rodiny AVR a to konkrétně ATmega8. Zde je popsán jak samotný procesor,tak i použitý periferní obvod jednotky pro sériovou komunikaci. Ve třetí kapitole je řešena samotná realizace bezdrátového ovládání. Uvedeno je také řešení desky plošných spojů pomocí programu EAGLE, tak i řešení programové psané v jazyce C. Poslední částí je závěr, kde jsou shrnuty mé poznatky a cíle této bakalářské práce.

2

Teoretický rozbor

2.1 Vysílače a legislativa ČR

Vysílání na volných frekvencích se v ČR řídí Všeobecným oprávněním VO-R/16/08.2005-28 k využívání rádiových kmitočtů a k provozování zařízení provozovaných společně na určených kmitočtech v pásmech 27 MHz až 450 MHz.

Provozní kmitočty	Vyzářený výkon	Šířka pásma zabraného vysíláním
27,905; 27,915 MHz	1 W	8,5 kHz
45,050; 45,175 MHz	1 W	16kHz
77,025; 78,000 MHz	1 W	16kHz
149,125; 149,250 MHz	500 mW	16kHz
172,650; 172,950 MHz	5 W	10kHz
305,825; 305,875 MHz	500 mW	16kHz
448,070; 448,170 MHz	500 mW	14kHz
449,770; 449,810 MHz	1 W	14kHz

Tab. 2.1: Tabulka některých frekvencí určených pro volné vysílání

Nejčastěji se používají moduly na frekvencích 434MHz, 868MHz a 2.4 GHz. V mém případě použiji moduly, pracující na frekvenci 433,92 MHz. Tato frekvence by měla být dostatečná vzhledem k mým požadavkům.

2.2 Technologie vysílače

Pro realizaci dálkového ovládání pro vjezdovou bránu, jsem se rozhodl použít některou z jednodušších technologií. Důvodem je to, že se bude přenášet poměrně malý datový tok, tudíž nepotřebuji ani velkou přenosovou rychlost. Vyplývá to ze vztahu 2.1 a 2.2 (kde "m" znamená počet použitých stavu). Těmto parametrům vyhovuje technologie amplitude-

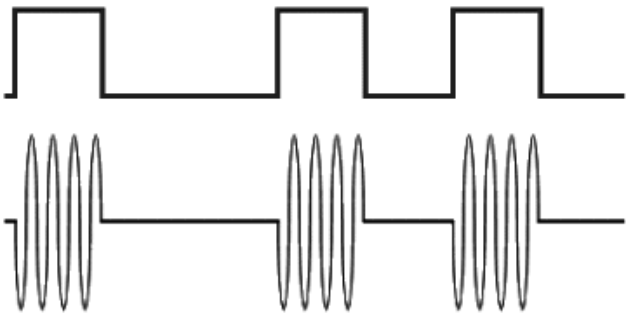
swift keying (ASK). Ta je založená na změně velikosti amplitudy podle vstupující binární kombinace. Bohužel jako každá amplitudová modulace je ovlivněná atmosférickými poruchami. Pro mé účely bude postačovat i nejjednodušší verze ASK a to konkrétně On-Off keying (OOK).

$$v_m = \frac{1}{a}, \quad \text{a} = \text{Doba trvání jednoho stavu.} \tag{2.1}$$

$$v_p = v_m \cdot log_2 m \tag{2.2}$$

2.2.1 On-Off keying

Jak již bylo zmíněno výše jedná se o nejjednodušší metodu digitální modulace signálu. Funguje tak, že pokud chceme vysílat signál o hodnotě logické "1", tak se vysílá sinusový puls o určené délce a frekvenci, pokud naopak vysíláme logickou "0", nevysílá se nic. Z výše uvedeného vyplývá i jeden nedostatek této technologie. Pokud přijímáme "0", tak si nemůžeme byt jistí, jestli byla opravdu vysílána "0", nebo se jen signál nevysílá. Funkce vysílače je ilustrována na obrázku 2.1, kde nahoře je signál, který chceme vysílat a dole je podoba vysílaného signálu. Modulátor je velice jednoduchý, skládá se pouze z oscilátoru a elektronického vypínače, který propouští signál do antény. Jako demodulátor se používá laděný obvod, který je naladěný na určitou frekvenci a za ním je paralelně připojen kondenzátor s rezistorem. Toto zapojení odstraní z největší části nosnou vlnu a tak zbyde jen pozvolna se měnící signál. Tento signál je potřeba ještě dále upravit. Tato úprava je prováděna pomocí Schmittova klopného obvodu, ten obnoví hrany signálu do původní podoby. Nyní je signal zrekonstruován a připraven k dalšímu zpracování.

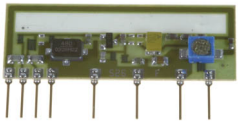


Obr. 2.1: ON-OFF keying

2.2.2 Modul vysílače

Jako vysílač byl použit modul TX-SAW I.A 433.92 MHz s integrovanou anténou od společnosti Enika.cz. Nosný signál moduluje digitálním signálem a využívá již výše zmíněnou technologii OOK. Pracovní frekvence je dána SAW rezonátorem, který kmitá na frekvenci 433,92 MHz. Modul dokáže pracovat s napájecím napětím 4V - 12 V. Výstupní výkon se

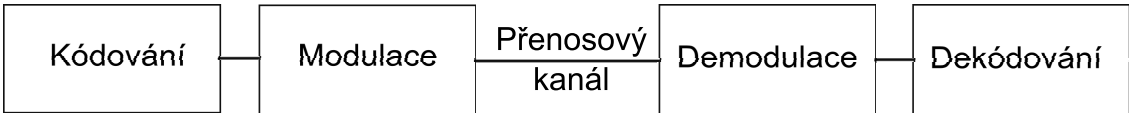
odvíjí od napájecího napětí. Při napájení 5V je 1mW a při 12V 2mW. Úroveň vstupního logického signálu je maximálně 5V a frekvence maximálně 4kHz. Odstup signálu od šumu je -40dB. Modul je zobrazen na obrázku 2.2



Obr. 2.2: Modul vysílače. Vývody jsou popsány zleva doprava. (zem,modulační vstup, modulační vstup, zem, zem, zem, zem, napájení)

2.3 Kódování signálu

Při přenosu informací dochází k chybám nejčastěji v přenosovém kanálu viz. obr. 2.3. Z tohoto důvodu se do vysílaného řetězce zavádí redundance, která nepřenáší žádnou informaci, ale dají se pomoci ní detekovat případné chyby a v některých případech se tyto chyby mohou i opravit.



Obr. 2.3: Základní části přenosového řetězu.

2.3.1 Hammingovo kódování

Jednou z metod kódování je Hammingovo kódování, které generuje vysílané slovo pomocí násobení informačního slova a generující matice. Pro určení správného kódu je potřeba si určit:

- Chybovost v použitém kanále, kterou zjistíme měřením.
- Podle chybovosti určit detekční schopnost algoritmu. Rovnice 2.3.
- Podle detekční schopnosti určit samoopravnost kódu. Rovnice 2.4.

$$D = d - 1, \quad \text{d} = \text{Vzdálenost mezi slovy generující matice} \tag{2.3}$$

$$O = \frac{D}{2}, \quad \text{O se zaokrouhluje na nižší celá čísla} \tag{2.4}$$

Jako ukázku použiji Hamm. kód (7,4). Odeslání Informačního slova "I" by vypadalo takto.

$$I = (0110).$$

Generující matice

$$G = \left(\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right)$$

Kódové slovo získáme použitím rovnice 2.5

$$C = I * G \tag{2.5}$$

$$C = (0110011)$$

Pro ilustraci opravení chyby, pozměníme přijaté slovo ve druhém bitu.

$$C' = (0010011)$$

Ke zpětnému dekódování se využívá matice H^T, která je tvořená pravou stranou generující matice a doplněna maticí jednotkovou.

$$H^T = \left(\begin{array}{ccc} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right)$$

Pro získání pozice chyby potřebujeme zjistit syndrom, který získáme podle rovnice 2.6.

$$S' = C' * H^T \tag{2.6}$$

$$S' = (101)$$

Takto získaný syndrom porovnáme s řádky matice H^T. V našem případě odpovídá syndrom druhému řádku. To znamená, že chyba nastala v druhém bitu a ten je tedy nutné opravit. Pokud syndrom neodpovídá s žádnému řádku a není-li nulový, nastala neopravitelná chyba. Pouze pokud je syndrom nulový je přijaté slovo kódové.

2.3.2 Perfektní kód

Perfektní kódy jsou takové kódy, kde mezi jednotlivými slovy jsou stejné mezery a vyplňují celý prostor 2.7.

$$q^n \geqq q^k \sum \binom{n}{k} (q-1)^k \tag{2.7}$$

Příklad perfektního kódu (3,1).

- pro 1 vyšleme (1 1 1)
- pro 0 vyšleme (0 0 0)

2.4 Technologie přijímače

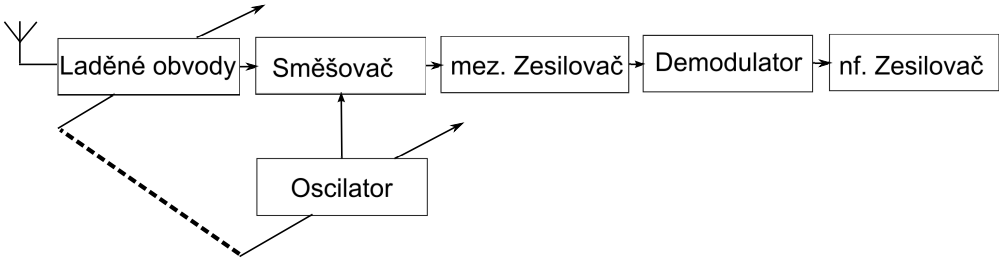
Typů přijímačů rádiových signálů je několik druhů. Je možné použít přímozesilující, který je sice konstrukčně jednoduchý, ale jeho selektivita je velmi malá. Ovšem nejčastěji používaným přijímačem je dnes superheterodyn nebo také superhet. Pracuje na principu přeložení přijímaného signálu do nižšího nebo vyššího pásma pomocí směšovače a to na tzv. mezifrekvenční signál. Toho dosahuje pomocí laděného obvodu a místního oscilátoru. Jelikož je jejich ladění spojeno, tak výsledkem sečtení nebo odečtení signálů, vzniká stále stejná frekvence.

Za směšovačem následuje mezifrekvenční zesilovač, který zesiluje získaný mezifrekvenční signál. Jeho konstrukce je taková, že je nastaven pouze na jednu frekvenci a to na frekvenci mezifrekvenční. Při směšování vzniká mezifrekvenční frekvence a zároveň mnoho dalších nežádoucích frekvencí. Ty nemají vliv na výsledný signál. Toho je docíleno tím, že mezifrekvenční zesilovač má poměrně velkou selektivitu.

Mnohdy je zapojeno i několik zesilovačů za sebou s vázanými rezonančními obvody pro zvýšení selektivity. Nevýhodou těchto přijímačů je zrcadlový příjem. Ten vzniká ve vf části přijímače a to tím, že pokud přijímáme signál na frekvenci f_a a signál f_b je vzdálený od signálu f_a o velikost mezifrekvence, tak následně budou ve směšovači po odečtení, přičtení tyto dva signály na stejné frekvenci. Odstranění zrcadlového přijmu se může dosáhnout, buď vhodným zvolením mezifrekvenčního kmitočtu (nejčastěji pro AM 450-460kHz a pro FM 10,7MHz), nebo více selektivním filtrem ve vstupním vf obvodu. Z tohoto vyplývá zajímavé zjištění, že pro zvýšení selektivity by bylo vhodné používat vyšší mezifrekvenční frekvenci, ale pro větší a lepší zesílení nf signálu používat mezifrekvenci nižší. Nejčastěji se problém řeší zlepšením selektivity vf obvodu.

2.4.1 Modul přijímače

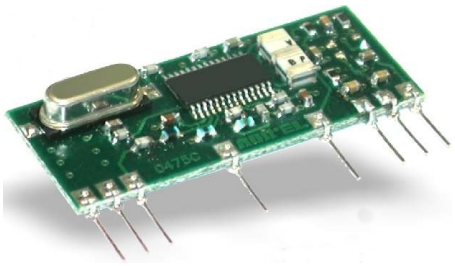
Přijímací modul RX 4MM3 AM super-het. Tento modul pracuje na frekvenci 433,92 MHz. Pro napájení je použito napětí 5V s odběrem 5,8mA. Dále má tento modul citlivost



Obr. 2.4: Blokové schéma přijímače typu heterodyn.

-114dBm. K jeho správné funkci je potřebné připojení antény. Anténa je volena podle přijímané frekvence jako zlomky vlnové délky. Její výpočet se provádí podle vzorce 2.8. Z této rovnice vychází délka antény 17,2cm. Podoba modulu přijímače je na obrázku 2.5.

$$\frac{\lambda}{4} = \frac{c}{f * 4} \tag{2.8}$$



Obr. 2.5: Modul přijímače (piny zleva N.C, zem, anténa, zem, AGC On-Off, test point, data out, napájení).

2.5 Programovací jazyk C

Je nízkoúrovňový multiplatformní programovací jazyk, který byl vyvinut v roce 1972 Dennisem Ritchiem pro potřeby operačního systému Unix. Tento jazyk vychází ze staršího assembleru, ale zjednodušuje syntaxi pro programování. Díky tomu, že každá platforma má vlastní překladač, umožňuje i jednoduchý port aplikací na jiné platformy. Tento jazyk podporuje i takzvaný inline assembler. To znamená, že přímo v části kódu je možné používat assembler. V dnešní době je tento jazyk používán téměř všude, od programování aplikací až po programování ovladačů a jader operačních systémů. Základní syntaktické prvky od něj přebraly i modernější programovací jazyky jako například Java, Perl, PHP a další.

3

Mikroprocesor Atmel AVR ATmega8

Jedná se o osmibitový procesor postavený jako RISC (angl. Reduced Instruction Set Computer). Technologie RISC se vyznačuje například tím, že se jedna instrukce provádí po dobu jednoho cyklu, využíváním víceúčelových registrů, nebo používáním instrukcí, které obsahují jak adresy obou zdrojů dat, tak i adresu kam se má výsledek po zpracování uložit.

Používá se také zřetězení instrukcí tzv. pipeline ve kterém jsou postupně načítány instrukce z paměti cache a v jednotlivých částech pipelinu a následně jsou postupně zpracovávány. Nejčastěji se používá pěti stupňový pipeline. To znamená, že se může zpracovávat až pět instrukcí najednou.

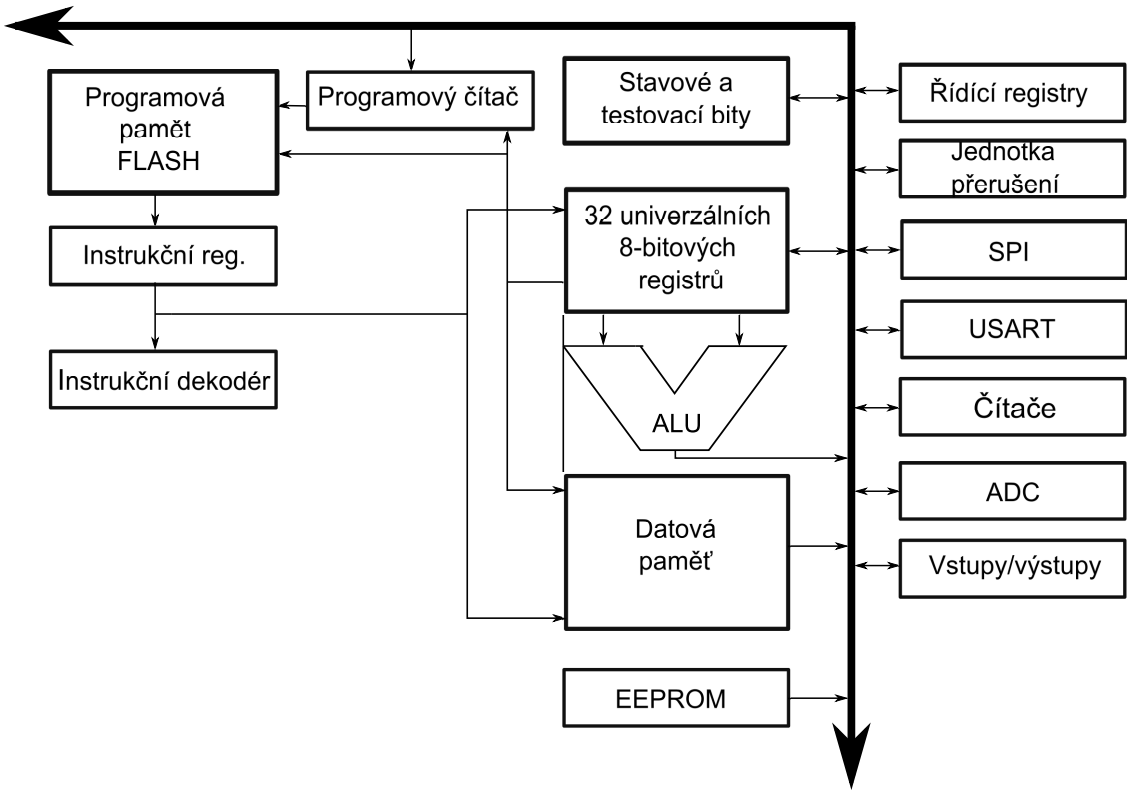
Tento princip se využívá ke zrychlení chodu procesoru, nevýhodou ale je problém vznikající při instrukci podmíněného skoku, při kterém se musí celý pipeline vyprázdnit. Zde také nastává problém, pokud dvě instrukce po sobě pracují se stejnými daty. V výsledek první operace ještě nemusí být připraven pro instrukci druhou.

Jádro procesorů AVR se skládá z 32 osmibitových registrů, do kterých se ukládají data i adresy. Tento procesor má oddělené paměťové prostory pro data a program, toto uspořádání se nazývá Hardwardská architektura. Zjednodušené blokové schéma je na obrázku 3.1

Parametry procesoru ATmega8

- 130 instrukcí procesoru
- 32 univerzálních 8 bitových registrů (GPR)
- 16 MHz maximální pracovní frekvence
- 8 KB flash paměti 10 000 přepsání
- 512 B EEPROM 100 000 přepsání
- 1 KB SRAM
- 2x 8-bit čítač/časovač

- 1x 16-bit čítač/časovač
- generátor reálného času s externím oscilátorem
- 3x PWM
- 10 bitový analog/digital převodník
- USART, SPI, TWI
- 5 režimů spánku
- 23 programovatelných vstupně/výstupních pinů
- Napájecí napětí 4,5 - 5,5 V (u modelu ATmega8L dokonce 2,7 - 5,5 V za cenu snížení pracovní frekvence na 8MHz)
- pouzdra PDIP, PQFP, MLF

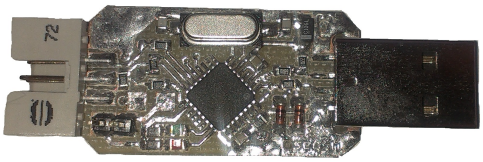


Obr. 3.1: Zjednodušené blokové schéma architektury AVR.

K programování mikroprocesoru jsou k dispozici 3 různé cesty. První z nich je paralelní programování, které se již v dnešní době moc nepoužívá. Důvodem je nemožnost programovat mikropočítač přímo v zapojení. Tato cesta se dá ale využít v případě špatného nastavení pojistek v procesoru, k jejich opravě, a tudíž i k záchraně samotného mikroprocesoru.

Druhý způsob je rozhraní JTAG, které kromě programování umožňuje i přímé ladění v zapojení. Program se pak může ladit ve vývojovém prostředí v počítači a zároveň je i možné používat tlačítka na hardwaru a vidět odezvu jak v registrech procesoru tak i na hardwaru.

Třetí a poslední možnost je kompromisem výše zmíněných, je to programování pomocí SPI. Jedná se o programování pomocí sériové komunikace, kde ke komunikaci slouží signály SCK, MISO a MOSI. Toto programování moderních procesorů je nejvíce rozšířeno a existuje mnoho druhů programátorů podporujících tento standard. Jsou to například programátory STK200, STK500 a USBASP který jsem použil já.

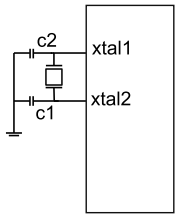


Obr. 3.2: Programátor USBASP.

3.1 Pojistky a generace času

Pro generaci hodinových impulsů pro procesor ATmega8 lze použít tři zdroje. Prvním zdrojem je vnitřní rezonanční obvod, který ale nemá dobrou kmitočtovou stabilitu a jeho použití jako zdroje hodinových impulsů se nedoporučuje. Používá se v podstatě pouze k programování procesoru.

Lze také využívat hodinové impulsy, které jsou generovány mimo procesor. K tomuto účelu jsou připraveny piny na pouzdře mikroprocesoru, označené jako XTAL1 a XTAL2. Na tyto piny lze přivést signál z vnějšího generátoru hodinových impulsu, to se používá pokud pracuje mikroprocesor jako součást většího zapojení, kde je potřeba aby byl tento celek synchronní. Poslední možností je připojení krystalového rezonátoru, to je nejpoužívanější možnost pro generování přesných hodinových signálů. Připojení krystalu k mikropočítači je znázorněno na obrázku 3.3.



Obr. 3.3: Zjednodušené schéma pro připojení oscilátoru.

Pro různé kmitočty se používají různé hodnoty kondenzátorů v rozmezí 12 - 22pF. V mém případě mám krystal o jmenovité frekvenci 12MHz. Pro tuto frekvenci je v datesheetu procesoru uvedena velikost kondenzátoru 22pF. Pro aktivaci vnějších hodinových

signálu z rezonátoru se dále musí nastavit pojistky, tzv. fuse bity. Ty se nastavují v dvojici registru, které určují jak se bude procesor chovat po resetu a co všechno bude dovoleno viz. tab 3.1. Při nastavování pojistek je potřeba dát pozor na správné nastavení hodin a pinu pro reset. Pokud budou špatně nastaveny hodiny procesor nebude schopen přijímat informace při sériovém programování a při nastavení pinu PC6(reset) jako vstupně/výstupní brány nepůjde sériově programovat vůbec. Pak se musí mikroprocesor vložit do paralelního programátoru a tam bity opravit. Bohužel to nejde u procesoru, které mají pouzdro provedeno jako smd a jsou již zapájené.

Název	Číslo bitu	Popis
RSTDISBL	7	Port PC0 je reset nebo I/O
WDTON	6	Zapnutí watchdogu
SPIEN	5	Povolí sériové čtení dat
CKOPT	4	Nastavení oscilátoru
EESAVE	3	EEPROM zůstává i přes smazání chipu
BOOTSZ1,0	2,1	Velikost boot sektoru
BOOTRST	0	nastavení vektoru resetu

Tab. 3.1: Tabulka pojistek v ATmega8 (horní bit)

Název	Číslo bitu	Popis
BODLEVEL	7	Nastavení úrovně brown outu
BODEN	6	Zapnutí detekce brown outu
SUT1	5	Nastavení startovacího času
SUT0	4	Nastavení startovacího času
CKSEL 3,2,1,0	3,2,1,0	Nastavení zdroje a tipu hodin

Tab. 3.2: Tabulka pojistek v ATmega8 (dolní bit)

V mém případě jsem pro nastavení hodin nastavil CKSEL3-0 na samé jedničky jak je vidět z následující tabulky 3.3

Nastavení zdroje hodin	CKSEL 3 - 0
Vnější krystal nebo keramický rezonátor	1111 - 1010
Vnější nízkofrekvenční krystal	1001
Externí RC oscilátor	1000 - 0101
Kalibrovaný vnitřní RC oscilátor	0100 - 0001
Externí hodiny	0000

Tab. 3.3: Tabulka nastavení CKSEL

Pro nastavení CKOPT je potřeba se podívat do tabulky módů, kde pro frekvence větší než 1MHz se nastaví bity CKSEL 3-1 na hodnotu logické jedničky a bity CKSEL 0 se

spolu s SUT1,0 se starají o zpoždění náběhu procesoru, z důvodu zpoždění rozkmitání krystalu na jmenovitou frekvenci. Hodnoty CKSEL0 a SUT1,0 nastavíme na 111 podle tab. 3.4, což odpovídá době zpoždění 65ms.

CKSEL0	SUT1,0	Doporučené použití
0	00	Keramický rezonátor, rychlý náběh
0	01	Keramický rezonátor, pomalý náběh
0	10	Keramický rezonátor, BOD povolen
0	11	Keramický rezonátor, rychlý náběh
1	00	Keramický rezonátor, pomalý náběh
1	01	Krystalový oscilátor, BOD povolen
1	10	Krystalový oscilátor, rychlý náběh
1	11	Krystalový oscilátor, pomalý náběh

Tab. 3.4: Tabulka nastavení CKSEL a SUT

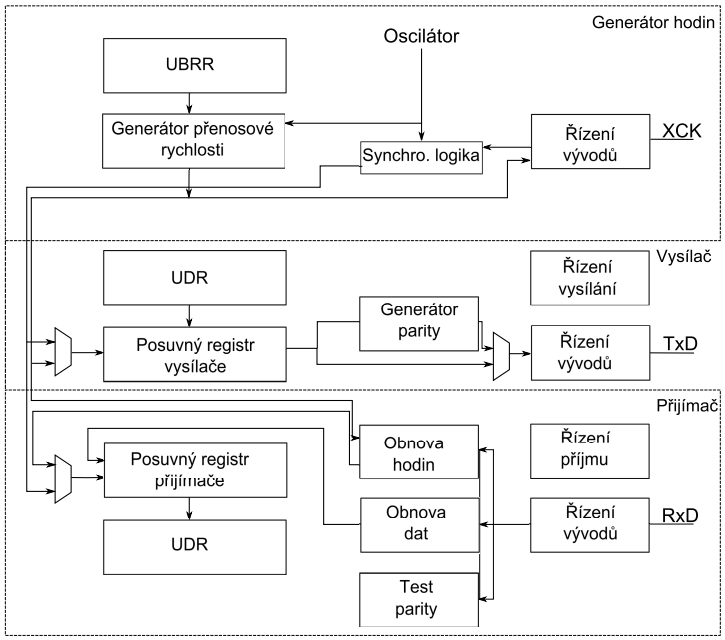
3.2 Univerzální synchronní asynchronní přijímač a vysílač (USART)

Rozhraní USART slouží pro sériovou komunikaci mezi různými zařízeními spadají sem i standardy jako je RS232 a RS485. Při použití asynchronního režimu se používá znaková synchronizace. To znamená, že se zasynchronizuje první přijatý bit, který se označuje jako start bit a pak se podle vnitřních hodin v pravidelném rytmu zachytí celé slovo. Na konci přenosu následuje stop bit, po kterém se může začít vysílat další znak. Tento protokol má klidový stav v log. 1 a odesílá standardně nejnižší bit slova jako první. U procesoru ATmega8 se o USART stará samostatná periferní jednotka.

Tato jednotka slouží pro velice flexibilní sériovou komunikaci, je plně duplexní tzn. že můžeme vysílat i přijímat data najednou. Dále tato jednotka umí vytvářet různé délky datových rámců a přidávat do nich stop bity i paritu, jak sudou tak lichou. Zvládá pracovat jak se synchronním tak i s asynchronním přenosem dat. Pro nastavení rychlosti je zde registr UBRR. K uložení přijatých dat se používá registr UDR, kam se data nasunou a dají o sobě vědět pomocí bitu RXC. Vysílání probíhá tak, že se data zapíšou do registru UDR a jsou ihned odvysílána. Odvysílání je potvrzeno signálem TXC. Před používáním sériové komunikace je ještě nutné povolit funkci bloků vysílače (nastavením TXEN) a přijímače (nastavením RXEN) na hodnotu log. "1".

Dále jsou zde další tři řídící a kontrolní registry UCSRA, UCSRB a UCSRC. Tyto registry se používají k nastavení činnosti USARTu a také se z nich získávají informace o přijatých zprávách jako například parita, devátý přenášený bit, pokud je nastaven rámec na devět znaků, chyba v přeneseném rámci a další.

Registr UCSRA = 00000000



Obr. 3.4: Zjednodušené blokové schéma pro USART.

- Bit 7 - RXC příjem je dokončen a data jsou připravena v UDR. Tento bit může vyvolat i přerušení.
- Bit 6 - TXC znak byl odvysílán a registr UDR je prázdný. Tento bit také může vyvolat přerušení.
- Bit 5 - UDRE se nastaví na 1 pokud je UDR prázdný.
- Bit 4 - FE chyba rámce zůstává nastaven dokud se nepřečte UDR.
- Bit 3 - DOR se nastaví pokud je UDR plný a přichází další start bit.
- Bit 2 - PE nastaví se pokud byla přijata chybná parita.
- Bit 1 - U2X nastavení dvojnásobné rychlosti.
- Bit 0 - MPCM nastavuje komunikaci mezi dvěmi procesory.

Registr UCSRB = 00011000

- Bit 7 - RXCIE povolení přerušení od přijímače.
- Bit 6 - TXCIE povolení přerušení od vysílače.
- Bit 5 - UDRIE povolení přerušení od prázdného zásobníku.
- Bit 4 - RXEN zapnutí přijímače.
- Bit 3 - TXEN zapnutí vysílače.

- Bit 2 - UCSZ2 spolu s UCSZ1,0 nastavuje formát vysílaného a přijímaného řetězce.
- Bit 1 - RXB8 slouží v případě odesílání rámce s 9-ti bity.
- Bit 0 - TXB8 slouží v případě odesílání rámce s 9-ti bity.

Registr UCSRC = 10000110

- Bit 7 - URSEL nastavuje čtení z UCSRC nebo z UBRRH
- Bit 6 - UMSEL nastavuje asynchronní (0) nebo synchronní (1) přenos
- Bit 5,4 - UPM1,0 nastavují paritu
- Bit 3 - USBS nastavení jednoho nebo dvou stop bitů
- Bit 2,1 - UCSZ1,0 nastavují datový rámec
- Bit 0 - UCPOL nastavení jestli bude systém reagovat na vzestupnou a nebo sestupnou hranu

Já jsem použil nastavení rámce na 8 bitů a jeden stop bit. Pro přenos dat jsem zvolil rychlost 9600 baudu, z čehož vyplývá nastavení UBRR na 77 podle následujícího vzorce 3.1.

$$UBRR = (\frac{f_{osc}}{16 \cdot 9600}) - 1 \tag{3.1}$$

Protokol USART se velmi často používá pro ladění činností procesoru, protože se dá lehce připojit do počítače pomoci sériového portu, díky čemu může mikroprocesor komunikovat s PC. Musí se ovšem myslet i na to, že standard RS232 využívá jiné napěťové úrovně a tak je potřeba použít vhodný převodník, jako je například max232. Tento převodník má v sobě zabudovanou nábojovou pumpu, která se stará o zvýšení napětí na ± 15V, pro dosažení standardu RS232. Tento standard má logickou "0" v +3V až v +15V a logická "1" je od -3V do -15V.

4

Realizace

4.1 Eagle

Pro navrhování desek plošných spojů jsem využil pro studenty volně dostupný program Eagle (Easily Applicable Graphical Layout Editor) ve verzi 6.3.0, vyvinutý firmou CadSoft. U verze pro studenty je program omezen na tvorbu DPS pouze do rozměru 100x80mm a jsou dostupné pouze dvě signálové vrstvy.

Tento program se skládá ze tří hlavních částí, a to řídícího panelu ve kterém se dají přidávat a ubírat různé knihovny součástek. Části pro navrhování zapojení a části pro návrh DPS.

Desky plošných spojů se v tomto programu navrhují tak, že se v části pro návrh zapojení přidají součástky jako schématické značky, u nichž jsou definovány i jejich pouzdra. Po tom se tyto součástky propojí do požadovaného zapojení. Tento program umožňuje i pojmenovat vodič jménem a pak budou všechny vodiče s tímto jménem propojeny. Toto se dá využít například pokud by propojení vodičů celé schéma znepřehlednilo a nebo k propojování různých bloků schématu. Po dokončení návrhu se nakonec ještě provede obvodová kontrola zapojení, kde se kontroluje jestli někde nejsou zkraty a nebo jsou některé důležité piny nezapojené. Tyto piny jsou například napájení nebo zem. Po dokončení návrhu zapojení následuje další část programu a to je návrh DPS.

Zde jsou již vidět součástky v pouzdrech a jsou mezi nimi vidět tzv. gumové spoje, které znázorňují, jak mají být součástky propojeny. Nyní se součástky přesouvají a umísťují na DPS. Po umístění součástek je zde možnost automatického propojení součástek, ale tato metoda je v tomto programu poměrně neefektivní a zbytečně využívá více vrstvé desky. Proto je lepší si propojení mezi obvody navrhnout ručně. Při propojování jsem používal ohyby vodičů o 45° z důvodu omezení rušení vznikajícího v ostrých rozích.

Po dokončení propojení, je potřeba se rozhodnout pro typ výrobního procesu. Pro výrobu pomocí kyseliny a leptání už není potřeba nic upravovat. Já jsem ale z důvodu výroby DPS na fréze, musel rozlýt po DPS měď, aby se frézovalo co nejméně. Tato rozlitá měď se nejčastěji přejmenovává na GND, aby se následně spojila se všemi zeměmi v zapojení. Nicméně pokud se zde nacházejí nepřipojené ostrůvky mědi, tak ty je potřeba

odstranit. Dále následuje ohraničení DPS čárou ve vrstvě milling, která určuje, kde se bude DPS odřezávat z původní desky.

Jako poslední věc je nutné zkontrolovat pravidla pro frézování (DRC). Do těchto pravidel patří dodržení izolačních mezer, kontroluje se velikost děr a další. Pokud tato kontrola proběhla v pořádku, je možné pokračovat k vytváření gerber souborů pro frézu a pro vrtačku. Jestli že kontrola v pořádku nebyla, je potřeba uvedené nedostatky odstranit.

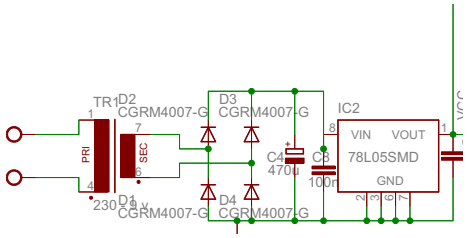
4.2 Napájení modulů

Pro napájení přijímacího modulu jsem zvolil řešení s vlastním transformátorem a usměrňovačem. Toto řešení je realizováno pomocí transformátoru, který transformuje napětí z 230V na primární straně na 9V na straně sekundární. Za tímto transformátorem je sestaven z diod usměrňovací můstek, jenž je realizován čtveřicí diod CGRM4007-G. Vzhledem k tomu, že modul bude odebírat minimální proud tak i diody, jako i ostatní součástky mohou být dimenzovány pro tyto nižší proudy. Očekávané odebírané proudy jsou kolem desítek miliampér. Za můstkem je elektrolytický kondenzátor s kapacitou 470 uF, který zde bude sloužit k pokrývání mezer mezi komutací diod. Opět vzhledem k proudům bude stačit, viz vzorec pro výpočet kapacity kondenzátoru 4.1.

$$C = \frac{I_m}{2 * f * \Delta U} \tag{4.1}$$

Z této rovnice po dosazení (za proud 90 mA, frekvenci 50Hz a Delta U 2V) vychází velikost kondenzátoru 450uF. Rozmezí ΔU může být i takto poměrně veliké hlavně díky tomu, že za usměrňovacím blokem je napětí 7V, a pokud bude ΔU do 2 voltů tak to stále neovlivní další obvody. Takže s tímto napěťovým rozsahem, by neměl být problém. O případné malé nadpětí se zde postará stabilizátor napětí 87L05SMD, na kterém také bude malý úbytek napětí kolem jednoho voltu, ale výslednou hodnotu napětí to nijak neovlivní. K tomuto stabilizátoru jsou po stranách připojeny malé kondenzátory. Dále jak si můžete všimnout ve schématu zapojení, jsem použil malé kondenzátory také před každým napájecím přívodem do jednotlivých součástek a to z toho důvodu, aby v případě zvýšeného odběru, byl tento proud odebírán prvotně z těchto kondenzátorů a neovlivnil zbytek zařízení.

U vysílače je napájení řešeno jednodušeji, jelikož se zde počítá s bateriovým zdrojem napětí, to znamená, že se nemusí nic usměrňovat a je zde pouze stabilizátor napětí LM2931M - 5.0. Na tomto stabilizátoru je oproti 78L05SMD menší úbytek napětí. Byl zvolen z důvodu menší napěťové rezervy ze zdroje. Jako zdroj jsem použil dvě knoflíkové baterie CR2032 s napětím 3V zapojené do série. Výsledné napětí tedy bude 6V. Další změnou je vypínač mezi baterii a stabilizátorem, který v případě sepnutí připojí baterii k zařízení.



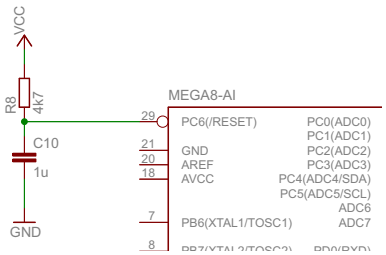
Obr. 4.1: Napájecí část.

4.2.1 Nulování procesoru (Reset)

Po náběhu napájecího napětí je nutné uvést procesor do definovaného stavu a k tomu se používá Reset. Reset může vyvolat hned několik podnětů, ale v mém případě využiji pin na pouzdře mikroprocesoru.

Tento reset je aktivní v nule, to znamená, že pokud přivedu na tento pin hodnotu log. "0", tak se procesor z resetuje. Jak můžete vidět na obrázku 4.2, je použito řešení s automatickým resetem po náběhu napájení. Je to vlastně RC článek, který se po připojení napájecího napětí začne nabíjet a po určité době se kondenzátor nabije nad rozhodovací úroveň a reset přestane být aktivní. Procesor pak může začít normálně fungovat a vykonávat instrukce programu. Doba, kterou trvá nabíjení kondenzátoru, je dána vztahem 4.2. Z tohoto vztahu po dosazení pro vysílač s kapacitou kondenzátoru 100nF vychází, že nabití na rozhodovací úroveň, která je v případě technologie CMOS v polovině napájecího napětí, tedy 2,5V bude trvat 0,325ms. Procesor ale v případě studeného startu potřebuje čas pouze 65ms, takže je zde dostatečná rezerva. U přijímače je tato rezerva ještě větší z důvodu ještě většího kondenzátoru.

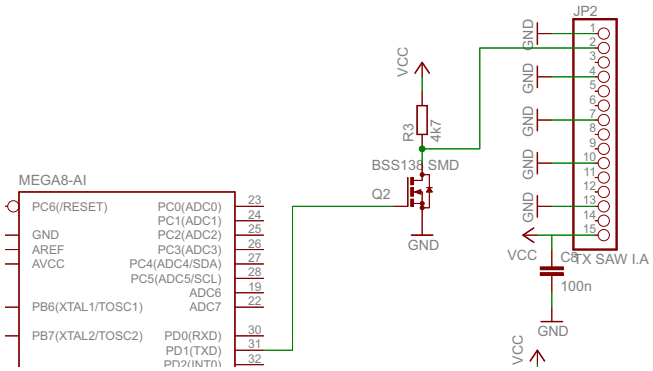
$$t = \tau \cdot \frac{U_c(t) - U_0}{U_0} \tag{4.2}$$



Obr. 4.2: Zapojení automatického resetu.

4.3 Připojení modulů k procesoru

Pro připojení modulu k procesoru bylo potřeba ještě vyřešit to, že modul vysílače vysílá v log "0" a v log "1" ne. Což neodpovídá protokolu pro USART u procesoru, kde je klidová hodnota 1. Bohužel tento procesor nepodporuje možnost negování výstupu USARTu přímo a tak bylo potřeba vytvořit hardwarovou negaci. Tato negace je k vidění na obr 4.3. Negace je tvořena unipolárním tranzistorem FET (BSS138SMD) a odporem proti napájecímu napětí. U přijímače je problém vyřešen obdobně akorát je gate tranzistoru připojen k modulu přijémače.



Obr. 4.3: Zapojení negace.

4.4 Software

Pro tvorbu programu jsem použil vývojové prostředí Atmel studio ve verzi 6.1.2440, které je volně dostupné. Toto prostředí je nadstavbou pro Visual studio od společnosti Microsoft. Toto vývojové prostředí vydává přímo společnost Atmel a soustřeďuje se v něm hlavně na procesory ARM a AVR pro které jsou zde připraveny jak softwarové debugery, tak i debugery pro rozhraní JTAG. Na začátku samotného programu volám knihovny, které jsou nutné pro správnou funkci programu, definuji zde frekvenci krystalu, rychlost přenosu, vzorec pro výpočet a nastavení přenosové rychlosti a jsou zde také nadefinovány bloky pro práci s výstupy. Dále se zde nachází definice proměnných.

```
1
2  #include <avr/io.h>
3  #include <stdio.h>
4  #define FOSC 12000000
5  #define BAUD 9600
6  #define MYUBRR (FOSC/(16*BAUD))-1
7  #define setb(port,pin)    port |= 1<<pin    //nastav bit
8  #define clrb(port,pin)    port &= ~(1<<pin) //nuluj bit
9  #define negb(port,pin)    port ^= 1<<pin    //neguj bit
10 typedef unsigned char byte;
11 unsigned char loop_cnt, len, cnt;
```

```
12 char heslo[] = {"ahoj"};
13
```

Následuje blok pro inicializaci sériového kanálu, kde je u přijímače nastaven pro příjem RXEN = 1 a u vysílače TXEN = 1 v registru UCSRB. Také se v tomto bloku nastavuje přenosová rychlost a definuje se podoba vysílaného, nebo přijímaného rámce.

```
1  void USART_init (unsigned int ubrr){
2
3          /* set baud rate*/
4          UBRRH = (unsigned char)(ubrr>>8);
5          UBRRL = (unsigned char)(ubrr);
6
7          /* enable receiver*/
8          UCSRB = (1<<TXEN);
9
10         /*set format*/
11         UCSRC = (1<<URSEL)|(3<<UCSZ0);
12 }
```

Dalším blokem je blok pro vysílání. Tento blok čeká dokud se předchozí znak celý neodvysílá a pak do registru pro vysílání (UDR) přiřadí další vysíany znak.

```
1  void USART_Transmit(unsigned char data){
2          /*wait for empty transmit buffer*/
3          while(!(UCSRA & (1<<UDRE)))
4                  ;
5          UDR = data;
6  }
```

U přijímače smyčka čeká dokud nebude znak přijat a když je přijat tak hodnotu znaku vrátí jako proměnnou char.

```
1
2  unsigned char USART_receive(void){
3          /*wait for receive*/
4          while(!(UCSRA & (1<<RXC)))
5                  ;
6          return UDR;
7  }
```

U vysílače následuje hlavní část programu, začínající nastavením potřebných portů jako výstupní, inicializací USARTu a hlavní smyčkou. Hlavní smyčka se stará o vysílání hesla. To probíhá tak, že po stisku tlačítka a počátečním resetu se rozsvítí LED dioda, zjistí se počet znaků v heslu a začnou odesílat postupně všechny znaky. Heslo se odešle celkem desetkrát poté dioda zhasne. Po zhasnutí diody se program zacykluje a čeká na ukončení v podobě uvilnění tlačítka.

```
1      clrb(PORTB,0);
2          len = strlen(heslo);
3          while(1)
4          {
5                          for(loop_cnt = 0; loop_cnt<10; loop_cnt++)
6                          {
7                                  for (cnt=0; cnt<len; cnt++)
8                                  {
9                                          USART_Transmit(heslo[cnt]);
10                                 }
11 }
12                 setb(PORTB,0);
13                 while(1);
14         }
```

U přijímače se v hlavní části programu čeká na první přijatý znak a pokud odpovídá prvnímu znaku hesla tak se začnou kontrolovat i další znaky. Pokud ne, tak se stále čeká na první shodný znak. Po přijetí tří správných hesel program vyhodnotí přijetí hesla jako úspěšné a dioda se na okamžik rozsvítí. Po úspěšném přijetí hesla je zde vložen blok, který zde čeká určitou dobu, aby byly ignorovány další přijímané znaky jako ochrana před tím, aby během jednoho cyklu vysílání hesla nebylo vyhodnoceno uspěšné přijmut vice než jednou.

```
unsigned char USART_receive(void){
len = strlen(heslo);

                if (USART_receive() == heslo[0]){
                for(loop=1; loop<len; loop++){
                        letter = USART_receive();
                                if (letter!=heslo[loop])
                                {
                                        valid = 0;
                                        break;
                                }
                                else;
                                {
                                        valid ++;
                                }
                        }
                if (valid >= 3){
                setb(PORTD,2);

                for (i=0; i<60000; i++)
                {
                        ;
                }
                clrb(PORTD,2);
                valid = 0;
                }}
```

4.5 Budoucnost

Pro budoucí rozšíření jsou na modulu přijímače připraveny výstupy pro připojení dalších součástí pro dokončení zbytku ovládání brány. Je zde připraven senzor pro Hallovu sondu, pomocí které budu měřit počet otáček ozubeného kola pro zjištění polohy brány a to tak, že na ozubené kolo připevním 2 magnety a s každým pulsem od Hallovy sondy budu vědět kolikrát se kolo otočilo. Tato sonda bude připojena na vstup procesoru PD2, který je také vstupem vnějšího přerušení. Toho se bude dát využít pro přičítání otáček i v případě, že se program bude zabývat něčím jiným. Dále je zde připraven konektor pro fotozávoru, která bude sloužit k odhalování překážek v prostoru brány. Dalším bezpečnostním prvkem bude měření proudu odebíraného motorem a v případě zvýšeného odběru může být tento stav vyhodnocen jako náraz do předmětu v prostoru brány. Tato metoda bude společně s čidlem polohy (Hallova sonda) sloužit také k řešení dojezdů brány. Toto měření proudu budu pravděpodobně realizovat pomocí Hallovy sondy připojené k A/D převodníku. Jeden pin v konektoru pro motor je také připojen ke generátoru PWM a tak bude možné regulovat rychlost otáček motoru.

5

Závěr

Cílem bakalářské práce bylo navrhnout a realizovat bezdrátové ovládání pro vjezdovou bránu. Při návrhu jsem dospěl k názoru, že pro toto řešení bude dostačovat použití poměrně jednoduchých technologii. Pro přenosovou část jsem použil technologii On - Off klíčování, která se vyznačuje svojí jednoduchostí a snadnou realizací. Pro příjem signálu jsem se rozhodl použít přijímač typu superhet, který se vyznačuje dobrou citlivostí. Jelikož se také jedná o nejrozšířenější typ přijímače, tak jeho sehnání nebyl problém. Pro realizaci vysílání a přijímání bylo potřeba vysílač a přijímač vhodně řídit.

Pro tuto činnost jsem zvolil mikroprocesor AVR ATmega8 jelikož, s ním mam již předchozí zkušenosti a také proto, že pro tuto činnost bez problémů dostačuje. K vysílání hesla jsem použil jednotku mikroprocesoru USART, která je schopná samostatně vysílat a přijímat znaky. Zde jsem narazil na problém s moduly pro příjem a vysílání. Tento problém se týkal negovaných vstupů a výstupů. To znamená, že nebylo možné vysílat přímo pomocí protokolu USART. Tento problém byl vyřešen hardwarovou negací tvořenou FET tranzistorem. Pro návrh desky plošných spojů jsem využil návrhové prostředí Eagle, které je schopné navrhnout kompletně celé zařízení od schématu, přes DPS, až po gerber soubory pro frézu. Zde se vysytl také malý problém, a to nekompletní knihovny součástek. Některé součástky se mi podařilo najít na webových stránkách výrobce jako přídavné knihovny, ale některé bylo potřeba vytvořit.

Výsledkem práce jsou dva moduly, které jsou schopné dálkového ovládání vjezdové brány. Modul přijímače je připraven pro další rozšíření v podobě fotozávory a modulu pro řízení motoru brány.

Literatura

[1] Pinker, Jiří. *Mikroprocesory a mikropočítače.* Praha: BEN, 2004. ISBN 80-7300-110-1.

[2] Hrabovký Miroslav, Juránek Antonín. *EAGLE pro začátečníky.* BEN, 2007. ISBN 80-7300-213-2

[3] Herout, Pavel. *Učebnice jazyka C.* KOOP:České Budějovice, 1992.

[4] ATmega8 *Katalogový list.* Atmel corporation:San Jose (California), 2008.

[5] Elweb.cz *Návrh stejnosměrného zdroje napětí[online].* Dostupné z www.elweb.cz/clanky.php?clanek=65.

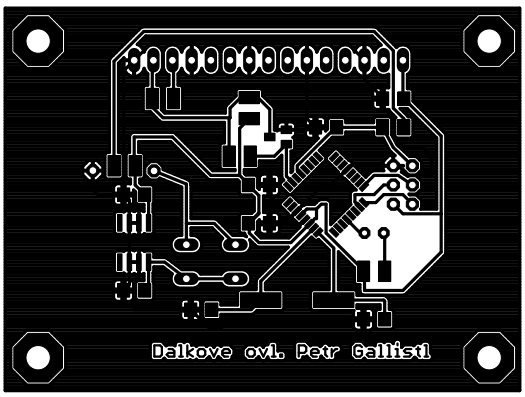
[6] Dlabos.wz.CZ *Rádiový přijímač-superhet[online].* Dostupné z www.dlabos.wz.cz/en/Radiove_prijimace.html.

[7] pandatron.cz *Antény pro malé bezdrátové systémy[online].* Dostupné z "www.pandatron.cz/?2888&anteny_pro_male_bezdratove_systemy."

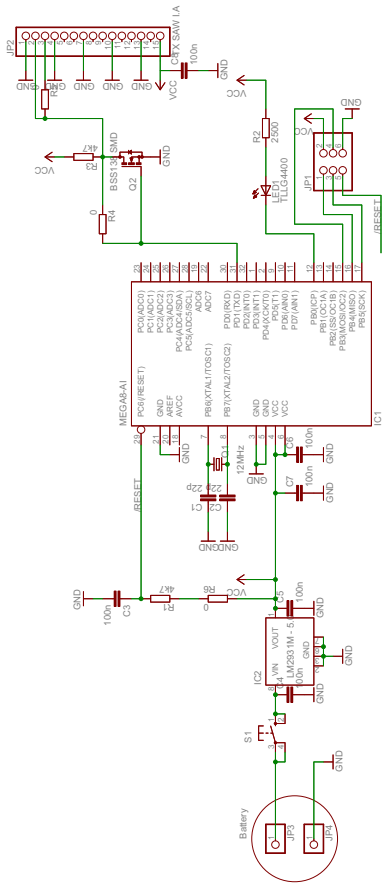
6

Desky plošných spojů, a schémata zapojení

6.1 Modul vysílače

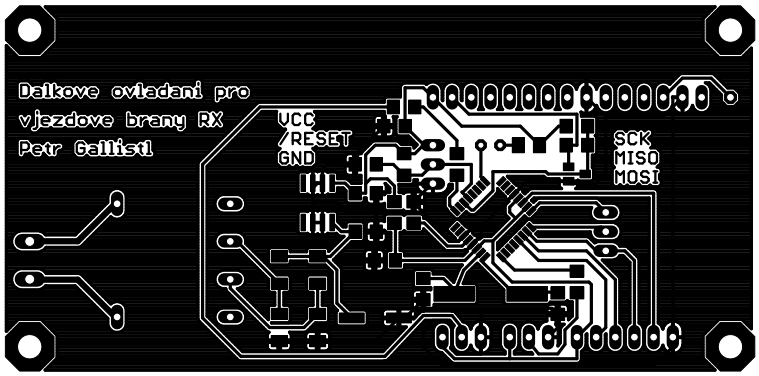


Obr. 6.1: DPS vysílače

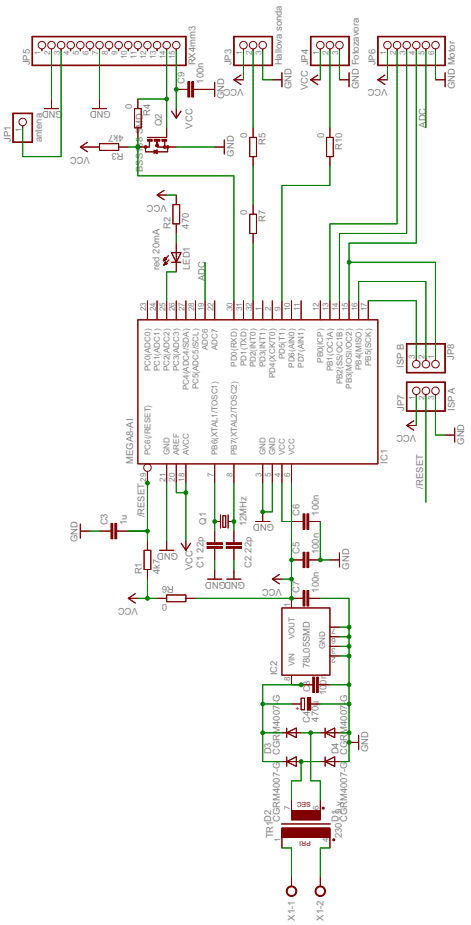


Obr. 6.2: Schéma vysílače

6.2 Modul přijímače



Obr. 6.3: DPS přijímače



Obr. 6.4: Schéma přijímače

7

Použité zdrojové kódy

7.1 Program vysílače

```
function diference = synchronizace(referencename,soundname,Nstart,Nstop,vykresleni)




#include <avr/io.h>
#include <stdio.h>
#define FOSC 12000000
#define BAUD 9600
#define MYUBRR (FOSC/(16*BAUD))-1
#define setb(port,pin)    port |= 1<<pin    //nastav bit
#define clrb(port,pin)    port &= ~(1<<pin) //nuluj bit
#define negb(port,pin)    port ^= 1<<pin    //neguj bit
typedef unsigned char byte;
unsigned char loop_cnt, len, cnt;
char heslo[] = {"ahoj"};


void USART_init (unsigned int ubrr){

        /* set baud rate*/
        UBRRH = (unsigned char)(ubrr>>8);
        UBRRL = (unsigned char)(ubrr);

        /* enable receiver*/
        UCSRB = (1<<TXEN);

        /*set format*/
        UCSRC = (1<<URSEL)|(3<<UCSZ0);

}

void USART_Transmit(unsigned char data){
        /*wait for empty transmit buffer*/
        while(!(UCSRA & (1<<UDRE)))
                ;
        UDR = data;
}

int main(void)
{
        PORTB = (1<<PB0);
        DDRB = (1<<DDB0);
//        USART_init(MYUBRR);
        USART_init(77);
        clrb(PORTB,0);
        len = strlen(heslo);
        while(1)
        {
```

```
50                          for(loop_cnt = 0; loop_cnt<10; loop_cnt++)
51                          {
52                                  for (cnt=0; cnt<len; cnt++)
53                                  {
54                                          USART_Transmit(heslo[cnt]);
55                                  }
56
57                          }
58
59                  setb(PORTB,0);
60                  while(1);
61          }
62  }
63
64
```

7.2 Program přijímače

```
1   #include <avr/io.h>
2   #define FOSC 12000000
3   #define BAUD 9600
4   #define MYUBRR (FOSC/(16*BAUD))-1
5   #define setb(port,pin)    port |= 1<<pin    //nastav bit
6   #define clrb(port,pin)    port &= ~(1<<pin) //nuluj bit
7   #define negb(port,pin)    port ^= 1<<pin    //neguj bit
8   char letter, heslo[] = {"ahoj"}, len, loop, pismeno;
9   typedef unsigned char byte;
10  byte valid;
11  unsigned int i = 0;
12
13
14
15  void USART_init (unsigned int ubrr){
16
17          /* set baud rate*/
18          UBRRH = (unsigned char)(ubrr>>8);
19          UBRRL = (unsigned char)(ubrr);
20
21          /* enable receiver*/
22          UCSRB = (1<<RXEN);
23
24          /*set format*/
25          UCSRC = (1<<URSEL)|(3<<UCSZ0);
26
27  }
28
29  unsigned char USART_receive(void){
30          /*wait for receive*/
31          while(!(UCSRA & (1<<RXC)))
32                  ;
33          return UDR;
34  }
35
36  int main(void)
37  {
38          PORTD = (1<<PD2);
39          DDRD = (1<<PD2);
40          setb(PORTD,2);
41
42  //        USART_init(MYUBRR);
43          USART_init(77);
44
45      while(1)
46      {
47          len = strlen(heslo);
48
49                  if (USART_receive() == heslo[0]){
50                  for(loop=1; loop<len; loop++){
51                          letter = USART_receive();
52                                  if (letter!=heslo[loop])
```

```
53                                      {
54                                              valid = 0;
55                                              break;
56                                      }
57                                      else;
58                                      {
59                                              valid ++;
60                                      }
61                              }
62                      if (valid >= 3){
63                      setb(PORTD,2);
64
65                      for (i=0; i<60000; i++)
66                      {
67                              ;
68                      }
69                      clrb(PORTD,2);
70                      valid = 0;
71                      }
72
73
74
75                      }
76      }
77  }
78
79
```