Středoškolská technika 2019

Setkání a prezentace prací středoškolských studentů na ČVUT

SIGNALIZACE VOZIDEL IZS PRO ŘIDIČE

Jan Mareš, Viktor Vitver

Střední škola průmyslová, textilní a polygrafická

Hostovského 910, Hronov

Poděkování

Rádi bychom touto cestou poděkovali Ing. Petru Neumannovi za velkou podporu, kterou nám poskytl při tvorbě této práce. Předal nám mnoho svých znalostí a zkušeností, především při volbě modulů, psaní programu a realizaci zařízení.

Dále bychom rádi poděkovali panu řediteli Ing. Josefu Matyášovi za myšlenku zpracování řešení této problematiky.

Prohlášení

Prohlašuji, že jsem svou práci vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Hronově dne ……

Jan Mareš

Viktor Vitver

Zadání:

Navrhněte a vytvořte prototyp zařízení (2 moduly) pro detekci vozidel s právem přednosti v jízdě, které lze připojit k běžnému vybavení automobilu. Zařízení musí umožnit optickou i akustickou signalizaci vozidla s právem přednosti v jízdě, které se pohybuje v okolí minimálně 100m a musí být schopno rozlišit minimálně 3 druhy těchto vozidel. Přenos signálu bude rádiový, přitom je nutné respektovat předpisy pro radiokomunikaci (Český telekomunikační úřad a České radiokomunikace). Zařízení se bude aktivovat automaticky při spuštění majáku, v ostatních vozidlech bude aktivní stále. Zadání doporučuji řešit pomocí mikroprocesoru a vhodného vf modulu. Při návrhu a realizaci využijte v maximální míře znalosti z výuky, zejména z předmětů Automatizace a řízení a Praxe.

Anotace:

Práce se zabývá problémem na silnicích, kdy řidič zaznamená vozidla s právem přednosti v jízdě velice pozdě. Bylo vytvořeno elektronické zařízení řízené mikroprocesorem, které by takové vozidlo zaznamenalo na vzdálenost cca 200m ve volném terénu. Vysílač by obsahovala vozidla s právem přednosti v jízdě, přijímač všechna ostatní vozidla. Vysílač vysílá data obsahující kód vozidla, přijímač přijatá data dekóduje a signalizuje přítomnost příslušného vozidla v okolí opticky i akusticky. Pro přenos dat byla použita volná frekvence 433MHz. Prototyp zařízení byl testován s různými vysokofrekvenčními moduly a s několika typy antén. Dosažené výsledky byly uspokojivé, zařízení je plně funkční. Pro případné další využití je popsáno, jaké problémy provázely vývoj zařízení a jejich řešení. Dále jsou uvedena doporučení pro sériovou výrobu a upozornění na povinnou homologaci.

Obsah

1. Úvod

Tato práce se zabývá tématem problému na silnicích, se kterým se potýká většina řidičů, na tuto problematiku nás upozornil pan ředitel Ing. Josef Matyáš. Jde o to, že řidič při své jízdě zaznamená vozidla s právem přednosti v jízdě velice pozdě. Je to způsobeno tím, že vozidla neslyší - např. z důvodu nahlas hrajícího rádia, použití sluchátek, hlasitému motoru, silné akustické izolaci vozu apod. Nebo to způsobuje, že vozidla nevidí - např. z důvodu špatných povětrnostních podmínek, vysoké hustoty zástavby, členitosti terénu, mrtvým úhlem výhledu z vozu apod.

Jako možné řešení jsme zvolili vizuální a akustickou signalizaci přímo na palubní desce vozu, které by měl zaznamenat každý řidič ve svém zorném poli. Náš zkušební modul jsme uzpůsobili i na rozeznání druhu blížícího se vozidla.

Používání přístrojů na bezdrátový přenos pro použití bez povolení je limitováno dodržením stanovených frekvencí a maximálního výkonu. My jsme zvolili běžně využívané pásmo 433MHz.

Tabulka sdílených frekvencí

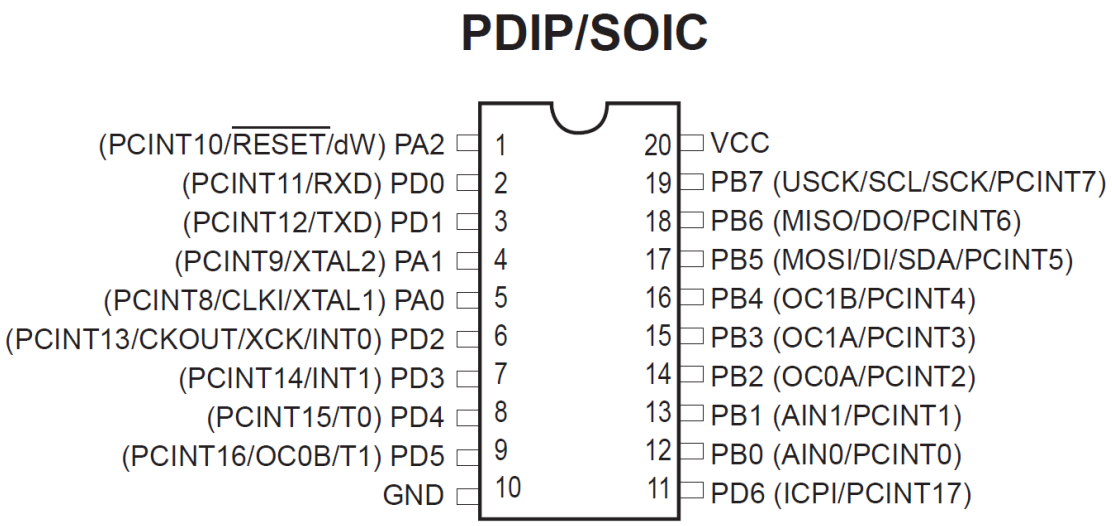
Tabulka 1

Kmitočtové pásmo (MHz)	Vyzářený výkon (mW)
40,660 - 40,700	10
138,200 - 138,450	10
433,050 - 434,790	10
863,000 - 870,000	25
2400 - 2483,5	25
5725 - 5875	25
24000 - 24250	100
61000 - 61500	100
122000 - 123000	100
244000 - 246000	100

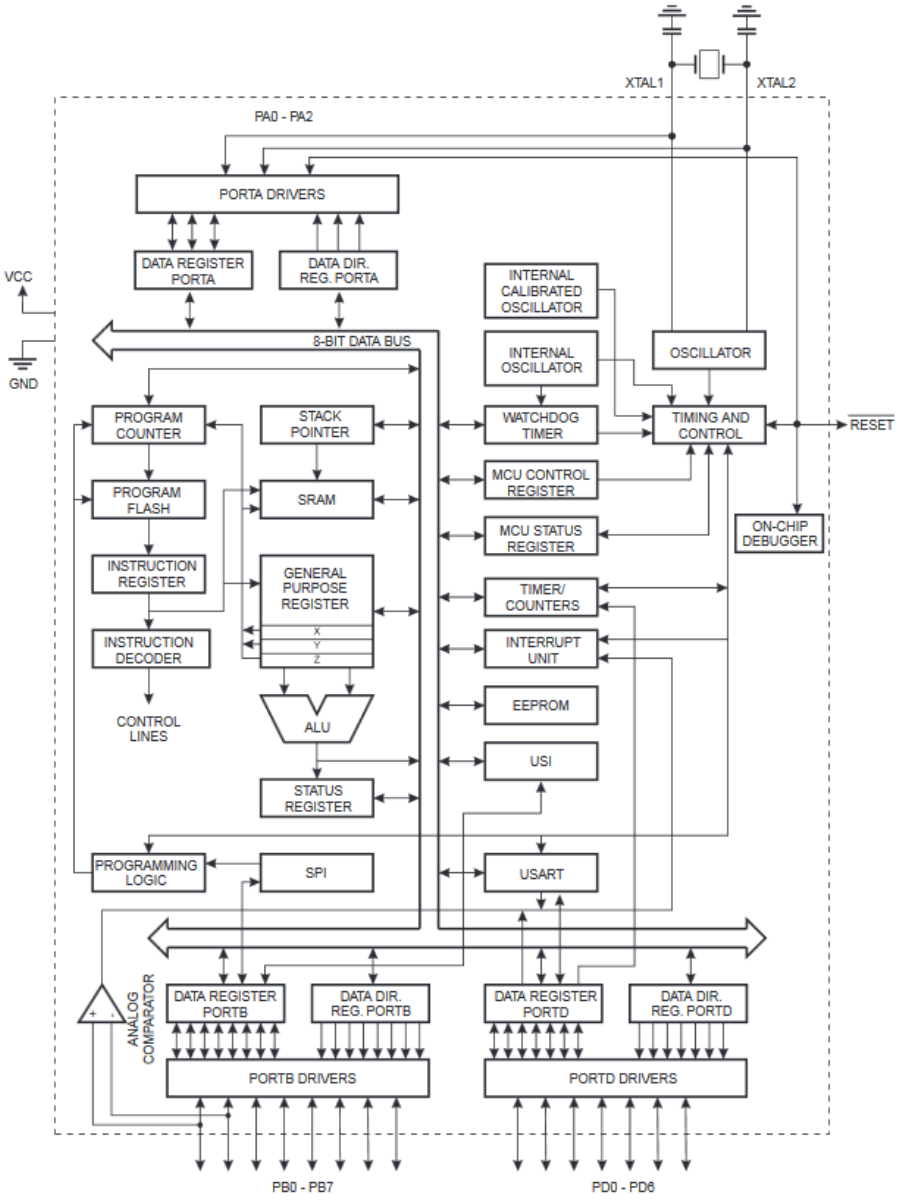
2. Výběr mikroprocesoru

Při výběru mikroprocesoru jsme zvolili mikroprocesor od výrobce Atmel, jelikož se ve výuce učíme programovat mikroprocesor ATmega32a, který je z řady AVR. Zvolili jsme provedení do patice, mikroprocesor jsme naprogramovali pomocí externího programátoru.

Jedná se o typ ATtiny4313 - výkonový 8 bitový mikroprocesor. Obsahuje 4KB FLASH, 256B SRAM, 128B EEPROM, 18 vstupně/výstupních linek a 32 univerzálních pracovních registrů. Dále obsahuje časovače, interní a externí přerušení, univerzální sériové rozhraní USART, programovatelný watchdog timer s interním oscilátorem a 3 režimy úspory energie. Pracovní napětí je od 1,8 do 5,5 voltů. Propustnost je 1 MIPS na MHz.



Obrázek 1 - vývodu mikroprocesoru



Obrázek 2 - periferie mikroprocesoru

Napájecí napětí modulů jsme zvolili 12V, protože to je běžné napětí v automobilech. 5V pro mikroprocesor jsme vytvořili standardním zapojením stabilizátoru 78L05. Aby bylo možné v průběhu vývoje testovat samotné moduly, doplnili jsme zapojení o jumpery nahrazující spínače a o signalizační led diody. Pro vf moduly jsou na deskách konektory.

3. Tvorba programu

Nejprve jsme se pokoušeli řešit komunikaci využitím integrovaného rozhraní USART, do přijímače však přichází takové množství různých signálů (jak jsme zjistili, dokonce i uprostřed lesa), že se v nich naše data zcela ztrácela. Proto jsme se po několika pokusech rozhodli zkusit použít open source licensing GPL V2 knihovnu VirtualWire. Ta je psána pro Arduino, Maple a další pro odesílání a přijímání krátkých zpráv bez adresování. Jelikož je univerzální museli jsme ji pro použití na našem mikroprocesoru nejprve zjednodušit (odstranit univerzální deklarace) aby byla přehlednější a pak upravit zbývající deklarace a některé funkce dle požadavků našeho programu.

Moduly umožňují teoreticky dosáhnout přenosové rychlosti až 9600 b/s. Vzhledem k nepřesnosti frekvence interního RC oscilátoru mikroprocesoru se projevovaly chyby komunikace již od rychlosti 4000 b/s. Proto jsme zvolili bezpečnou rychlost 1000 b/s, která pro naše potřeby bohatě stačí.

Data jsou vysílána jako řetězce znaků, které tvoří příslušný kód, funkce „vw_send“ z knihovny k těmto datům připojí hlavičku datového rámce a kontrolní součet, který je při přijetí dat funkcí z knihovny také kontrolován.

3.1. Program pro vysílač

Na začátku programu je vložena knihovna VirualWire.h, která obsahuje funkce potřebné pro komunikaci, zároveň knihovna využívá časovač T1, kterým se řídí přenosová rychlost. Dále je v programu využito přerušení od časovače T0, pomocí kterého se odměřuje interval 0,5s pro odesílání dat. Po spuštění programu se provede inicializace portů, nastavení přenosové rychlosti funkcí „vw_timerSetup“ z knihovny a nastavení výchozích hodnot našich proměnných.

V hlavní smyčce se nejprve zjišťují stavy vstupů, když je spuštěný maják, tak se podle dalších přepínačů nastavují hodnoty proměnných „data“ (kód vozidla) a „ledky“ (zobrazení stavu na výstupech). Přitom se zároveň nastaví proměnná „vysilej“ (start vysílání). Pokud je spuštěno vysílání, kontroluje se náběžná hrana signálů na vstupech mikroprocesoru od ovládacích spínačů a následně se spouští časovač T0, který provede první vyslání dat funkcí „vw_send“ z knihovny a následně čeká na odeslání dat pomocí funkce „vw_wait_tx“ z knihovny. V pravidelných intervalech 0,5s se poté opakuje odeslání dat. Po dobu, kdy se odesílají data, se zároveň rozsvítí příslušná ledka. Protože odeslání jednoho bitu trvá 1ms, je tato doba pro signalizaci funkce dostatečná.

3.2. Program pro přijímač

Program je napsán podobným způsobem a začíná vložením knihovny VirualWire.h. Dále se v programu nachází obsluha přerušení od časovače T0, pomocí kterého se odměřuje interval 0,25s. Po spuštění programu se provede inicializace portů, nastavení přenosové rychlosti funkcí „vw_timerSetup“ z knihovny, spuštění příjmu dat funkcí „vw_rx_start“ z knihovny a nastavení výchozích hodnot našich proměnných.

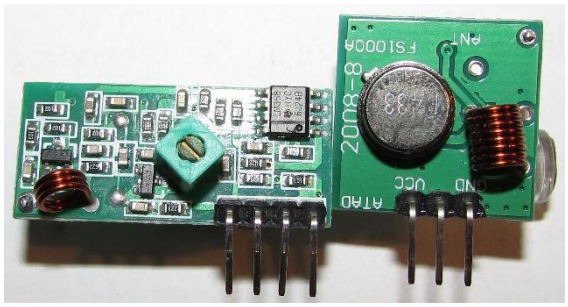
V hlavní smyčce se nejprve dekódují přijatá data (hasiči, sanitka, policie, rezerva) a pokud je zjištěn náš kód, rozsvítí se příslušná led dioda příchozího kódu a vynuluje se počítadlo, které zajišťuje dobu svícení led diody 3 s. Kontroluje se náběžná hrana příchozího kódu a při ní se spouští časovač T0. V pravidelných intervalech 0,25s se inkrementují počítadla svítících led diod a pokud led dioda svítí již 3s, tak se zhasne. Zároveň se v těchto intervalech neguje proměnná „zvuk“. Pokud svítí některá z led diod, tak se aktivuje port s piezo reproduktorem, který pravidelně píská, pokud nesvítí žádná led dioda, vypne se zvuk a časovač.

4. Výběr VF modulů

Nejprve jsme zvolili moduly Arduino 433 MHz, které jsou uzpůsobeny na komunikaci s vývojovými kity Arduino, Raspberyy PI opod. Z důvodu horších vlastností modulů Arduino jsme objednali moduly Aurel TX-SAW433/S-Z + RX BC-NBK, která mají mnohem lepší vlastnosti.

4.1. Arduino 433 MHz vysílač + přijímač

Moduly jsou stavěny pro komunikaci do vzdálenosti 200 m. Vzdálenost je ovlivněna přenosovou rychlostí, napájecím napětím vysílače a samozřejmě anténou a jejím připojením.



Obrázek 3

Přijímač Vysílač

Tabulka 2

Specifikace přijímacího modulu

Typ přijímače	XY-MK-5V	**Komunikační frekvence (MHz)**	315 nebo 433,92
Provozní napětí	5 VDC	**Citlivost (50 Ω)**	-100 dBm
Provozní proud	< 6 mA	**Přenos. rychlost (@ 315 MHz, -95 dBm)**	< 9,6 Kb/s
Rozměry (mm)	30 x 14	**Max. dosah modulů**	20–200 m

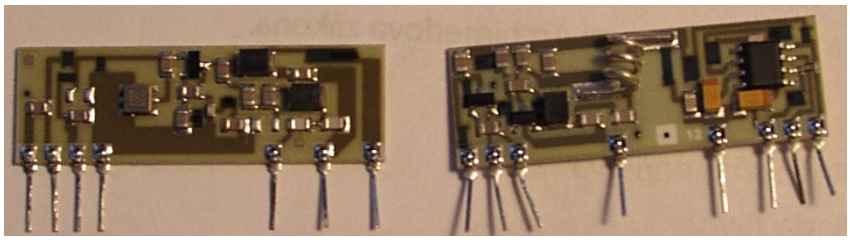
Tabulka 3

Specifikace vysílacího modulu

Typ vysílače	XY-FST	**Komunikační frekvence (MHz)**	315 nebo 433,92
Provozní napětí	3–12 VDC	**Vysílací výkon (@ 315 MHz, 12 V)**	25 mW
Provozní proud	9–40 mA	**Přenos. rychlost (@ 315 MHz, -95 dBm)**	< 9,6 Kb/s
Šířka pásma	2 MHz	**Rozměry (mm)**	19 x 19

4.2. Aurel TX-SAW433/S-Z + RX BC-NBK

Jedná se o vysílač se SAW rezonátorem pro modulaci nosné vlny digitálním signálem a VF přijímač s nízkou spotřebou a malým vyzařováním antény.



Obrázek 4

Vysílač Přijímač

Specifikace přijímacího modulu

Tabulka 4

VF výstup	zátěž 50 Ω – pin 11	Pracovní frekvence	433,92 MHz
Provozní napětí	5V	Anténa	λ/4
Provozní proud	3mA	Rozměry(mm)	38,1x13,7
Modulace	On-Off klíčování	Max. frekvence	je dána SAW rezonátorem

Vysoká odolnost proti rušení, vestavěný RC filtr

Specifikace vysílacího modulu

Tabulka 5

VF výstup	zátěž 50 Ω – pin 11	Pracovní frekvence	433,92 MHz
Provozní napětí	5V	Výstupní výkon	max. +10 dBm
Provozní proud	4mA	Rozměry(mm)	38,1x13,2
Pracovní teplota	–20 až +80 °C	Max. frekvence	je dána SAW rezonátorem

Je charakteristický vysokou spolehlivostí a nízkým rušivým zářením, vysoká spolehlivost SIL thick-film hybridního obvodu

4.3. Výběr antény

Antény jsme přilepili pomocí tavné pistole na destičky cuprextitu, abychom měření neovlivňovali svým tělem nebo jiným kontaktem.

4.3.1. Čtvrtvlnná anténa z vodiče

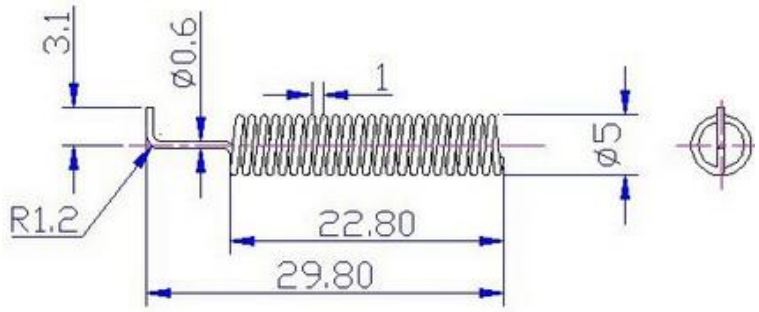
Anténu jsme vytvořili z měděného vodiče o délce

$$l = \frac{\frac{rychlost\ světla}{frekvence\ modulu}}{čtvrt\ vlna} = \frac{\frac{300000000}{433920000}}{4} = 0{,}173m = 173mm$$

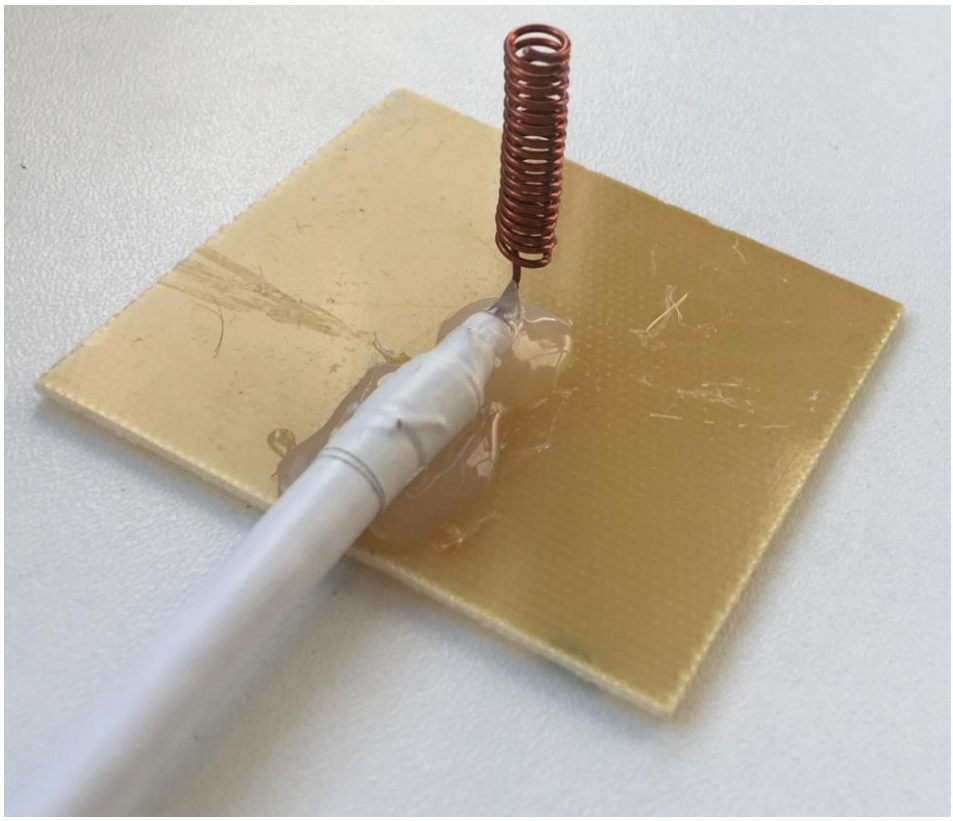
Obrázek 5

4.3.2. Anténa doporučená výrobcem

Dle zadání výrobce modulu jsme vyrobili anténu dle nákresu, která je velmi náchylná na přesnost výroby, chování v případě dodržení přesnosti by mělo být ideální.



Obrázek 6



Obrázek 7

4.3.3. Profesionální anténa z dálkového ovladače

Měli jsme ve škole k dispozici profesionální anténu na 869 MHz, sice to není frekvence, kterou používají naše moduly, ale zajímala nás funkčnost s tímto typem antény.

Obrázek 8

4.3.4. Bez antény

Pro zajímavost jsme zkusili, jaký mají moduly dosah bez antény, tj. vysílaly a přijímaly signál pouze pinem na připojení antény.

5. Testování zařízení

Protože nemáme ve škole dostatečné vybavení pro vf měření, museli jsme testování provádět praktickými pokusy. První pokusy probíhaly v zimních měsících venku, což samo o sobě přinášelo značné problémy (teploty pod nulou, sníh).

Největší potíže způsobilo to, že jsme nepoužili externí krystal, ale spolehli se na vnitřní kalibrovaný RC oscilátor mikroprocesoru. Jak jsme postupně zjistili (měřením osciloskopem a ochlazováním v mražáku), jeho frekvence není pro účely komunikace dostatečně přesná. Navíc je frekvence velmi závislá na teplotě, což se projevilo při testování zařízení venku při nízkých teplotách.

Dalším problémem byl vliv okolí na vf pole, při položení modulu s anténou na zem se dosah výrazně zkrátil, při držení modulu v ruce byl dosah ovlivněn tělem držícího.

Proto byly závěrečné pokusy provedeny v dlouhé chodbě školy, mikroprocesory jsme doplnily externími krystaly a moduly stály na podstavcích. Bohužel jsme nemohli zjistit dosah delší než délka chodby (50m). Proto jsme zkoušeli hlavně varianty s jedinou anténou nebo bez ní. Zde se projevil už jen jediný problém, nejlepší anténa při připojení k vysílači s modulem Arduino způsobovala rušení samotného modulu a komunikace selhávala, modul Aurel tento problém neměl.

Zjištěný dosah modulů v metrech

Tabulka 6

modul \ anténa	1	2	3	4
Arduino (vysílač bez antény)	28	24	19	12
Arduino (přijímač bez antény)	nezjištěno	přes 50	přes 50	12
Aurel (vysílač bez antény)	přes 50	přes 50	přes 50	16
Aurel (přijímač bez antény)	přes 50	přes 50	přes 50	16

6. Ovládání

Vzhled panelů se nachází v přílohách.

6.1. Vysílač

Vysílač obsahuje 2 panely, jeden s páčkami a konektorem na připojení antény a druhý se signalizačními diodami.

Panel s přepínači slouží k ovládání, prvním přepínačem zleva se vysílač aktivuje (zapne se maják) a dalšími se určuje vysílaný signál (druh vozidla), vždy jde vysílat maximálně jeden signál.

Na panelu s diodami slouží největší dioda k signalizaci aktivace zařízení (maják) a další znázorňují, který signál se vysílá.

6.2. Přijímač

Přijímač obsahuje 2 panely, jeden s přepínačem a konektorem na připojení antény a druhý se signalizačními diodami.

Přepínač slouží k aktivaci zvukové signalizace přijímaného signálu.

Na panelu s diodami vidíme, který signál přijímáme.

7. Programy

7.1. Knihovna VirtualWire (upravená)

Soubor vw.h:

```
#ifndef   VirtualWire_h
#define  VirtualWire_h

#define  F_CPU   8000000UL

#include <avr/interrupt.h>
#include <util/delay.h>
#include <string.h>
#include <stdbool.h>
#include  <stdint.h>

#define  VW_TX_PORT                  PORTD
#define  VW_TX_DDR                   DDRD
#define  VW_TX_PIN                   PIND1
#define  VW_RX_PORT                  PIND
#define  VW_RX_DDR                   DDRD
#define  VW_RX_PIN                   PIND0
#define  VW_RX_RAMP_LEN              160
#define  VW_RX_SAMPLES_PER_BIT       8
#define  VW_TIMSK                    TIMSK
#define  VW_MAX_MESSAGE_LEN          40
#define  VW_MAX_PAYLOAD              VW_MAX_MESSAGE_LEN - 3
#define  VW_RAMP_INC                 (VW_RX_RAMP_LEN/VW_RX_SAMPLES_PER_BIT)
#define  VW_RAMP_TRANSITION          VW_RX_RAMP_LEN / 2
#define  VW_RAMP_ADJUST              9
#define  VW_RAMP_INC_RETARD          (VW_RAMP_INC-VW_RAMP_ADJUST)
#define  VW_RAMP_INC_ADVANCE         (VW_RAMP_INC + VW_RAMP_ADJUST)
#define  VW_HEADER_LEN               8
#define  lo8(x)                      ((x) & 0xff)
#define  hi8(x)                      ((x) >> 8)

#define  impulzyT0 195; //0,02496s

extern    void vw_digitalWrite_tx(bool x);
extern    void vw_set_rx_inverted(uint8_t inverted);
extern    void vw_set_ptt_inverted(uint8_t inverted);
extern    void vw_pll();
extern    void vw_pinSetup();
extern    void vw_timerSetup(uint16_t speed);
extern    void vw_setup(uint16_t speed);
extern    void vw_tx_start();
extern    void vw_tx_stop();
extern    void vw_rx_start();
extern    void vw_rx_stop();
extern    void vw_wait_tx();
extern    void vw_wait_rx();
extern    bool vw_digitalRead_rx();
extern    uint8_t _crc_ibutton_update(uint8_t crc, uint8_t data);
extern    uint8_t vw_symbol_6to4(uint8_t symbol);
extern    uint8_t vw_timer_calc(uint16_t speed, uint16_t max_ticks, uint16_t *nticks);
extern    uint8_t vw_tx_active();
extern    uint8_t vw_wait_rx_max(unsigned long milliseconds);
extern    uint8_t vw_send(uint8_t* buf, uint8_t len);
extern    uint8_t vw_have_message();
extern    uint8_t vw_get_message(uint8_t* buf, uint8_t* len);
extern    uint8_t vw_get_rx_good();
```

```
extern    uint8_t vw_get_rx_bad();
extern    uint16_t crc16_update(uint16_t crc, uint8_t a);
extern    uint16_t crc_xmodem_update (uint16_t crc, uint8_t data);
extern    uint16_t _crc_ccitt_update (uint16_t crc, uint8_t data);
extern    uint16_t vw_crc(uint8_t *ptr, uint8_t count);

#endif
```

Soubor vw.cpp:

```
#include "VirtualWire.h"

#define  vw_digitalWrite_ptt(value)

static    uint8_t vw_tx_len = 0;
static    uint8_t vw_tx_index = 0;
static    uint8_t vw_tx_bit = 0;
static    uint8_t vw_tx_sample = 0;
static    uint8_t vw_ptt_inverted = 0;
static    uint8_t vw_rx_inverted = 0;
static    uint8_t vw_rx_sample = 0;
static    uint8_t vw_rx_last_sample = 0;
static    uint8_t vw_rx_pll_ramp = 0;
static    uint8_t vw_rx_integrator = 0;
static    uint8_t vw_rx_active = 0;
static    uint8_t vw_rx_enabled = 0;
static    uint8_t vw_rx_bit_count = 0;
static    uint8_t vw_rx_buf[VW_MAX_MESSAGE_LEN];
static    uint8_t vw_rx_count = 0;
static    uint8_t vw_rx_bad = 0;
static    uint8_t vw_rx_good = 0;
static    uint8_t vw_tx_buf[(VW_MAX_MESSAGE_LEN * 2) + VW_HEADER_LEN]
                = {0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x2a, 0x38, 0x2c};
static    uint8_t symbols[]
                = {0xd, 0xe, 0x13, 0x15, 0x16, 0x19, 0x1a, 0x1c, 0x23, 0x25, 0x26, 0x29, 0x2a, 0x2c, 0x32, 0x34};
static    volatile uint8_t vw_tx_enabled = 0;
static    volatile uint8_t vw_rx_done = 0;
static    volatile uint8_t vw_rx_len = 0;
static    uint16_t vw_tx_msg_count = 0;
static    uint16_t vw_rx_bits = 0;

void vw_digitalWrite_tx(bool x)
{
          if (x)
          {
                    VW_TX_PORT |= (1 << VW_TX_PIN);
          }
          else
          {
                    VW_TX_PORT &= ~(1 << VW_TX_PIN);
          }

}

bool vw_digitalRead_rx()
{
          if
          {
                    (VW_RX_PORT & (1 << VW_RX_PIN))
                    return true;
          }

          else
          {
                    return false;
          }
```

```
}

uint16_t crc16_update(uint16_t crc, uint8_t a)
{
	int i;
	crc ^= a;
	for (i = 0; i < 8; ++i)
	{
		if (crc & 1)
			crc = (crc >> 1) ^ 0xA001;
		else
			crc = (crc >> 1);
	}
	return crc;
}

uint16_t crc_xmodem_update (uint16_t crc, uint8_t data)
{
	int i;
	crc = crc ^ ((uint16_t)data << 8);
	for (i = 0; i < 8; i++)
	{
		if (crc & 0x8000)
			crc = (crc << 1) ^ 0x1021;
		else
			crc <<= 1;
	}
	return crc;
}

uint16_t _crc_ccitt_update (uint16_t crc, uint8_t data)
{
	data ^= lo8(crc);
	data ^= data << 4;
	return ((((uint16_t)data << 8) | hi8 (crc)) ^ (uint8_t)(data >> 4) ^ ((uint16_t)data << 3));
}

uint8_t _crc_ibutton_update(uint8_t crc, uint8_t data)
{
	uint8_t i;
	crc = crc ^ data;
	for (i = 0; i < 8; i++)
	{
		if (crc & 0x01)
			crc = (crc >> 1) ^ 0x8C;
		else
			crc >>= 1;
	}
	return crc;
}

uint16_t vw_crc(uint8_t *ptr, uint8_t count)
{
	uint16_t crc = 0xffff;
	while (count-- > 0)
		crc = _crc_ccitt_update(crc, *ptr++);
	return crc;
}

uint8_t vw_symbol_6to4(uint8_t symbol)
{
	uint8_t i;
	uint8_t count;
	for (i = (symbol >> 2) & 8, count = 8; count-- ; i++)
		if (symbol == symbols[i])
			return i;
	return 0;
}
```

```
void vw_set_rx_inverted(uint8_t inverted)
{
	vw_rx_inverted = inverted;
}

void vw_set_ptt_inverted(uint8_t inverted)
{
	vw_ptt_inverted = inverted;
}

void vw_pll()
{
	if (vw_rx_sample)
		vw_rx_integrator++;
	if (vw_rx_sample != vw_rx_last_sample)
	{
		vw_rx_pll_ramp += ((vw_rx_pll_ramp < VW_RAMP_TRANSITION)
			? VW_RAMP_INC_RETARD : VW_RAMP_INC_ADVANCE);
		vw_rx_last_sample = vw_rx_sample;
	}
	else
		vw_rx_pll_ramp += VW_RAMP_INC;
	if (vw_rx_pll_ramp >= VW_RX_RAMP_LEN)
	{
		vw_rx_bits >>= 1;
		if (vw_rx_integrator >= 5)
			vw_rx_bits |= 0x800;
		vw_rx_pll_ramp -= VW_RX_RAMP_LEN;
		vw_rx_integrator = 0;
		if (vw_rx_active)
		{
			if (++vw_rx_bit_count >= 12)
			{
				uint8_t this_byte = (vw_symbol_6to4(vw_rx_bits & 0x3f)) << 4
					| vw_symbol_6to4(vw_rx_bits >> 6);
				if (vw_rx_len == 0)
				{
					vw_rx_count = this_byte;
					if (vw_rx_count < 4 || vw_rx_count > VW_MAX_MESSAGE_LEN)
					{
						vw_rx_active = false;
						vw_rx_bad++;
						return;
					}
				}
				vw_rx_buf[vw_rx_len++] = this_byte;
				if (vw_rx_len >= vw_rx_count)
				{
					vw_rx_active = false;
					vw_rx_good++;
					vw_rx_done = true;
				}
				vw_rx_bit_count = 0;
			}
		}
		else if (vw_rx_bits == 0xb38)
		{
			vw_rx_active = true;
			vw_rx_bit_count = 0;
			vw_rx_len = 0;
			vw_rx_done = false;
		}
	}
}

uint8_t vw_timer_calc(uint16_t speed, uint16_t max_ticks, uint16_t *nticks)
{
	uint16_t prescalers[] = {0, 1, 8, 64, 256, 1024, 3333};
	uint8_t prescaler = 0;
```

```
	unsigned long ulticks;
	if (speed == 0)
	{
		*nticks = 0;
		return 0;
	}
	for (prescaler=1; prescaler < 7; prescaler += 1)
	{
		unsigned long inv_clock_time = F_CPU / ((unsigned long)prescalers[prescaler]);
		unsigned long inv_bit_time = ((unsigned long)speed) * 8;
		ulticks = inv_clock_time / inv_bit_time;
		if ((ulticks > 1) && (ulticks < max_ticks))
			break;
	}
	if ((prescaler == 6) || (ulticks < 2) || (ulticks > max_ticks))
	{
		*nticks = 0;
		return 0;
	}
	*nticks = ulticks;
	return prescaler;
}

void vw_pinSetup()
{
	VW_TX_DDR   |=  (1 << VW_TX_PIN);
	VW_RX_DDR   &= ~(1 << VW_RX_PIN);
}

void vw_timerSetup(uint16_t speed)
{
	uint16_t nticks;
	uint8_t prescaler;
	prescaler = vw_timer_calc(speed, (uint16_t)-1, &nticks);
	TCCR0A = 0b00000010;
	TCCR0B = 0b00000101;
	OCR0A = impulzyT0;
	TCCR1A = 0b00000000; //výchozí
	TCCR1B = 0b00001000;
	TCCR1B |= prescaler;
	TCCR1C = 0b00000000; //výchozí
	OCR1A = nticks;
	TIMSK = 0b01000000;
	sei();
}

void vw_setup(uint16_t speed)
{
	vw_pinSetup();
	vw_digitalWrite_ptt(vw_ptt_inverted);
	vw_timerSetup(speed);
}

void vw_tx_start()
{
	vw_tx_index = 0;
	vw_tx_bit = 0;
	vw_tx_sample = 0;
	vw_digitalWrite_ptt( true ^ vw_ptt_inverted);
	vw_tx_enabled = true;
}

void vw_tx_stop()
{
	vw_digitalWrite_ptt(false ^ vw_ptt_inverted);
	vw_digitalWrite_tx(false);
	vw_tx_enabled = false;
}
```

```
void vw_rx_start()
{
	if (!vw_rx_enabled)
	{
		vw_rx_enabled = true;
		vw_rx_active = false;
	}
}

void vw_rx_stop()
{
	vw_rx_enabled = false;
}

uint8_t vw_tx_active()
{
	return vw_tx_enabled;
}

void vw_wait_tx()
{
	while (vw_tx_enabled)
		;
}

void vw_wait_rx()
{
	while (!vw_rx_done)
		;
}

uint8_t vw_wait_rx_max(unsigned long milliseconds)
{
	while(milliseconds--)
	{
		if(vw_rx_done)
			break;
		_delay_ms(1);
	}
	return vw_rx_done;
}

uint8_t vw_send(uint8_t* buf, uint8_t len)
{
	uint8_t i;
	uint8_t index = 0;
	uint16_t crc = 0xffff;
	uint8_t *p = vw_tx_buf + VW_HEADER_LEN;
	uint8_t count = len + 3;
	if (len > VW_MAX_PAYLOAD)
		return false;
	vw_wait_tx();
	crc = _crc_ccitt_update(crc, count);
	p[index++] = symbols[count >> 4];
	p[index++] = symbols[count & 0xf];
	for (i = 0; i < len; i++)
	{
		crc = _crc_ccitt_update(crc, buf[i]);
		p[index++] = symbols[buf[i] >> 4];
		p[index++] = symbols[buf[i] & 0xf];
	}
	crc = ~crc;
	p[index++] = symbols[(crc >> 4) & 0xf];
	p[index++] = symbols[crc & 0xf];
	p[index++] = symbols[(crc >> 12) & 0xf];
	p[index++] = symbols[(crc >> 8) & 0xf];
	vw_tx_len = index + VW_HEADER_LEN;
	vw_tx_start();
	return true;
```

```
}

uint8_t vw_have_message()
{
	return vw_rx_done;
}

uint8_t vw_get_message(uint8_t* buf, uint8_t* len)
{
	uint8_t rxlen;
	if (!vw_rx_done)
		return false;
	rxlen = vw_rx_len - 3;
	if (*len > rxlen)
		*len = rxlen;
	memcpy(buf, vw_rx_buf + 1, *len);
	vw_rx_done = false;
	return (vw_crc(vw_rx_buf, vw_rx_len) == 0xf0b8);
}

uint8_t vw_get_rx_good()
{
	return vw_rx_good;
}

uint8_t vw_get_rx_bad()
{
	return vw_rx_bad;
}

ISR(TIMER1_COMPA_vect)
{
	if (vw_rx_enabled && !vw_tx_enabled)
		vw_rx_sample = vw_digitalRead_rx() ^ vw_rx_inverted;
	if (vw_tx_enabled && vw_tx_sample++ == 0)
	{
		if (vw_tx_index >= vw_tx_len)
		{
			vw_tx_stop();
			vw_tx_msg_count++;
		}
		else
		{
			vw_digitalWrite_tx(vw_tx_buf[vw_tx_index] & (1 << vw_tx_bit++));
			if (vw_tx_bit >= 6)
			{
				vw_tx_bit = 0;
				vw_tx_index++;
			}
		}
	}
	if (vw_tx_sample > 7)
		vw_tx_sample = 0;
	if (vw_rx_enabled && !vw_tx_enabled)
		vw_pll();
}
```

7.2. Vysílač

```
// program pro vysílač (z TxD = pin3 = D1 při stisknutí D6, jumpery D2 až D5, ledky B0 až B3)

#include <avr/io.h>
#include "VirtualWire.h"

// deklarace globálních proměnných
uint8_t pocetT0 = 0;
bool casT0 = false;

// přerušení od časovače T0
ISR(TIMER0_COMPA_vect) //0,02496s
{
	pocetT0++;
	if (pocetT0 == 20) //0,4992s
	{
		pocetT0 = 0;
		casT0 = true;
	}
}

// hlavní funkce programu
int main(void)
{

// inicializace vstupů a výstupů
	DDRD = 0b00000010;
	PORTD = 0b11111100;
	DDRB = 0b11111111;
	PORTB = 0b00000000;

	vw_timerSetup(1000);

// deklarace lokállních proměnných
	const char *data = "";
	uint8_t ledky = 0;
	bool vysilej = false;
	bool vysila = false;

// nekonečná smyčka pro vysílání dat
	while (1)
	{

// volba vysílaného kódu
		switch (PIND | 0b10000011)
		{
			case 0b10111011: //jumper D2 + D6
				data = "has"; //kód hasiči
				ledky = 1; // rozsvícení led B0
				vysilej = true;
				break;
			case 0b10110111: //jumper D3 + D6
				data = "san"; //kód sanitka
				ledky = 2; //rozsvícení led B1
				vysilej = true;
				break;
			case 0b10101111: //jumper D4 + D6
				data = "pol"; //kód policie
				ledky = 4; // rozsvícení led B2
				vysilej = true;
				break;
			case 0b10011111: //jumper D5 + D6
				data = "rez"; //kód rezerva
				ledky = 8; // rozsvícení led B3
				vysilej = true;
				break;
```

```
                default:
                        TIMSK &= 0b11111110;
                        vysila = false;
                        vysilej = false;
        }

// vysílání zvoleného kódu
        if (vysilej)
        {
                if (!vysila)
                {
                        vysila = true;
                        pocetT0 = 0;
                        casT0 = false;
                        TCNT0 = 0;
                        TIMSK |= 0b00000001;
                        PORTB = ledky;
                        vw_send((uint8_t *)data, strlen(data));
                        vw_wait_tx();
                        PORTB = 0;
                }

                if (casT0)
                {
                        casT0 = false;
                        PORTB = ledky;
                        vw_send((uint8_t *)data, strlen(data));
                        vw_wait_tx();
                        PORTB = 0;
                }
        }
    }
}
```

7.3.Přijímač

```
// program pro přijímač (do RxD = pin2 = D0, ledky B0 až B3, zvuk B4)

#include  <avr/io.h>
#include  "VirtualWire.h"

// deklarace globálních proměnných
uint8_t pocetT0 = 0;
bool casT0 = false;


// přerušení od časovače T0
ISR(TIMER0_COMPA_vect) //0,02496s
{
	pocetT0++;
	if (pocetT0 == 10) //0,2496s
	{
		pocetT0 = 0;
		casT0 = true;
	}
}

// hlavní funkce programu
int main(void)
{

// inicializace vstupů a výstupů
	DDRD = 0b00000010;
	PORTD = 0b11111100;
	DDRB = 0b11111111;
	PORTB = 0b00000000;

	vw_timerSetup(1000);
	vw_rx_start();

// deklarace lokállních proměnných
	uint8_t buf[VW_MAX_MESSAGE_LEN] = {0, 0, 0};
	uint8_t buflen = VW_MAX_MESSAGE_LEN;
	uint8_t pocetB0 = 0;
	uint8_t pocetB1 = 0;
	uint8_t pocetB2 = 0;
	uint8_t pocetB3 = 0;
	bool sviti = false;
	bool zvuk = false;

// nekonečná smyčka pro příjem dat
	while (1)
	{

// vyhodnocení přijatého kódu
		if (vw_get_message(buf, &buflen))
		{
			if ((buf[0] == 'h') && (buf[1] == 'a') && (buf[2] == 's')) //kód hasiči
			{
				PORTB |= 0b00000001; //ledka B0
				pocetB0 = 0;
			}
			if ((buf[0] == 's') && (buf[1] == 'a') && (buf[2] == 'n')) //kód sanitka
			{
				PORTB |= 0b00000010; //ledka B1
				pocetB1 = 0;
			}
			if ((buf[0] == 'p') && (buf[1] == 'o') && (buf[2] == 'l')) //kód policie
			{
				PORTB |= 0b00000100; //ledka B2
				pocetB2 = 0;
```

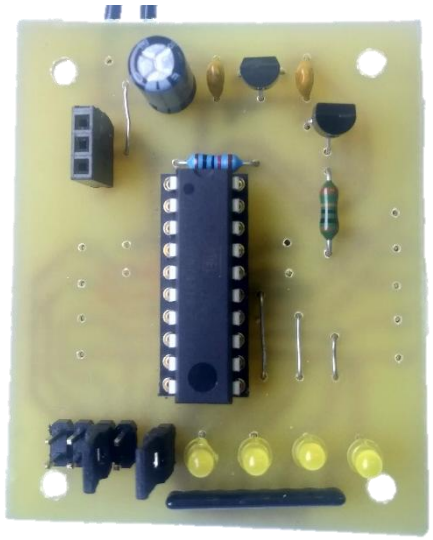
```
                        }
                        if ((buf[0] == 'r') && (buf[1] == 'e') && (buf[2] == 'z')) //kód rezerva
                        {
                                PORTB |= 0b00001000; //ledka B3
                                pocetB3 = 0;
                        }
                        if (!sviti)
                        {
                                sviti = true;
                                pocetT0 = 0;
                                casT0 = false;
                                TCNT0 = 0;
                                TIMSK |= 0b00000001;
                                zvuk = true;
                        }
                }

// prodleva před zhasnutím ledky + akustická signalizace
                if (casT0)
                {
                        casT0 = false;
                        pocetB0++;
                        pocetB1++;
                        pocetB2++;
                        pocetB3++;
                        if (pocetB0 == 12) //cca 3s
                                PORTB &= 0b11111110; //ledka B0
                        if (pocetB1 == 12) //cca 3s
                                PORTB &= 0b11111101; //ledka B1
                        if (pocetB2 == 12) //cca 3s
                                PORTB &= 0b11111011; //ledka B2
                        if (pocetB3 == 12) //cca 3s
                                PORTB &= 0b11110111; //ledka B3
                        zvuk = !zvuk;
                }

// ukončení signalizace
                if (!(PORTB & 0b00001111))
                {
                        sviti = false;
                        TIMSK &= 0b11111110;
                        zvuk = false;
                        PORTB &= 0b11101111; //ledka B4
                }
                else
                {
                        if (zvuk)
                                PORTB |= 0b00010000; //ledka B4
                        else
                                PORTB &= 0b11101111; //ledka B4
                }
        }
}
```

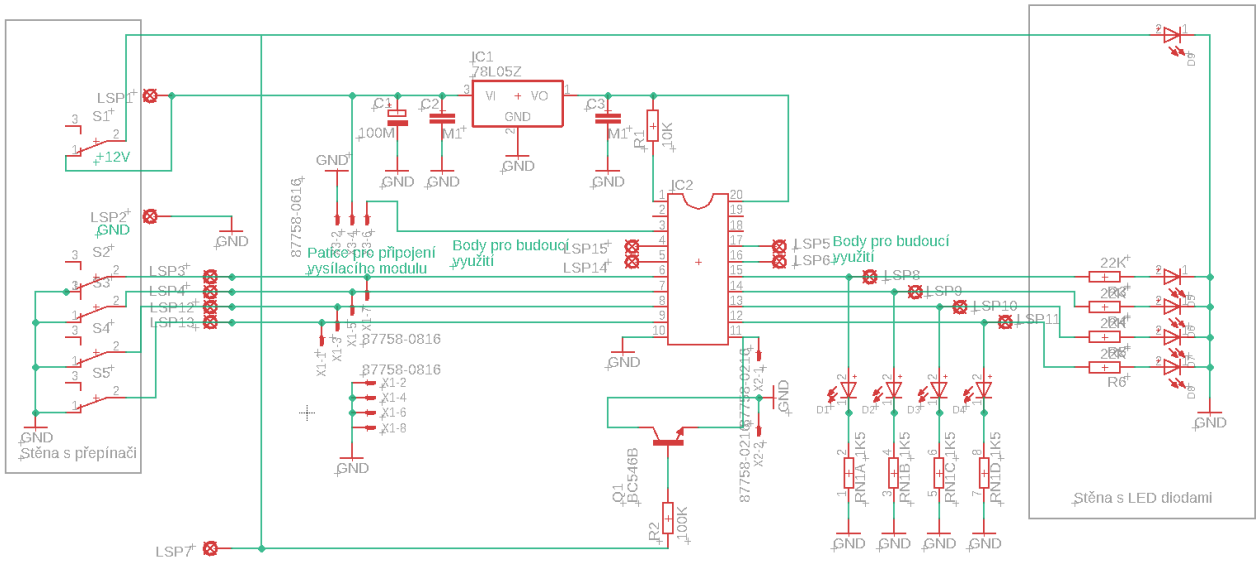
8. Přílohy

8.1. Vysílač



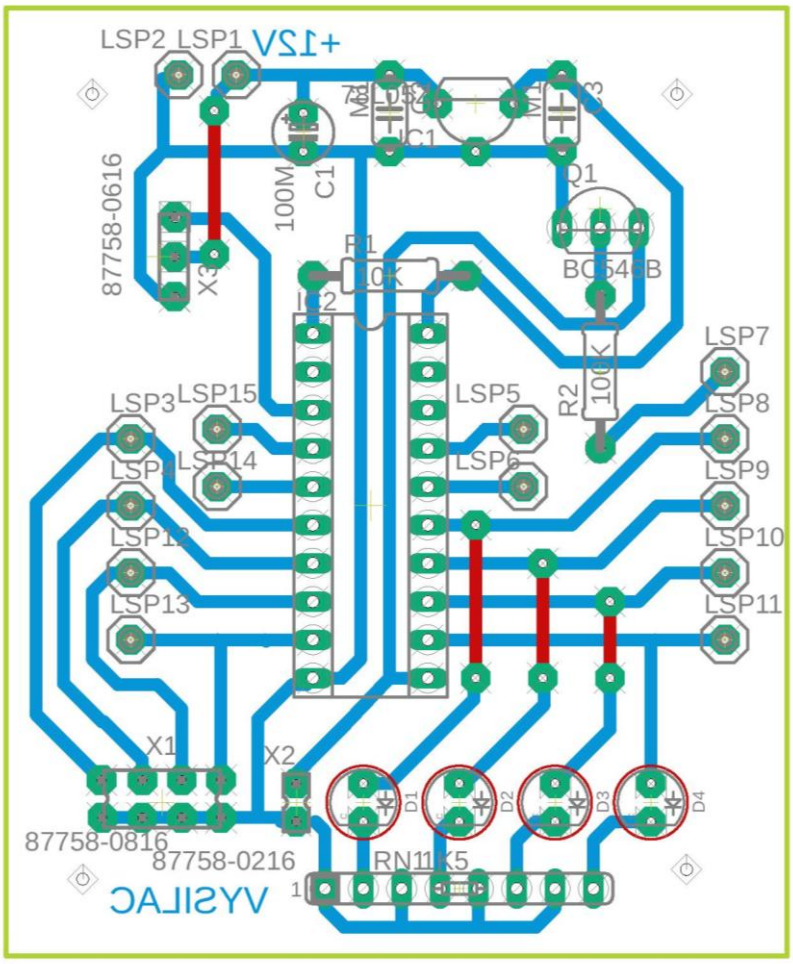
Obrázek 9

8.1.1. Schéma



Obrázek 10

8.1.2. Plošný spoj



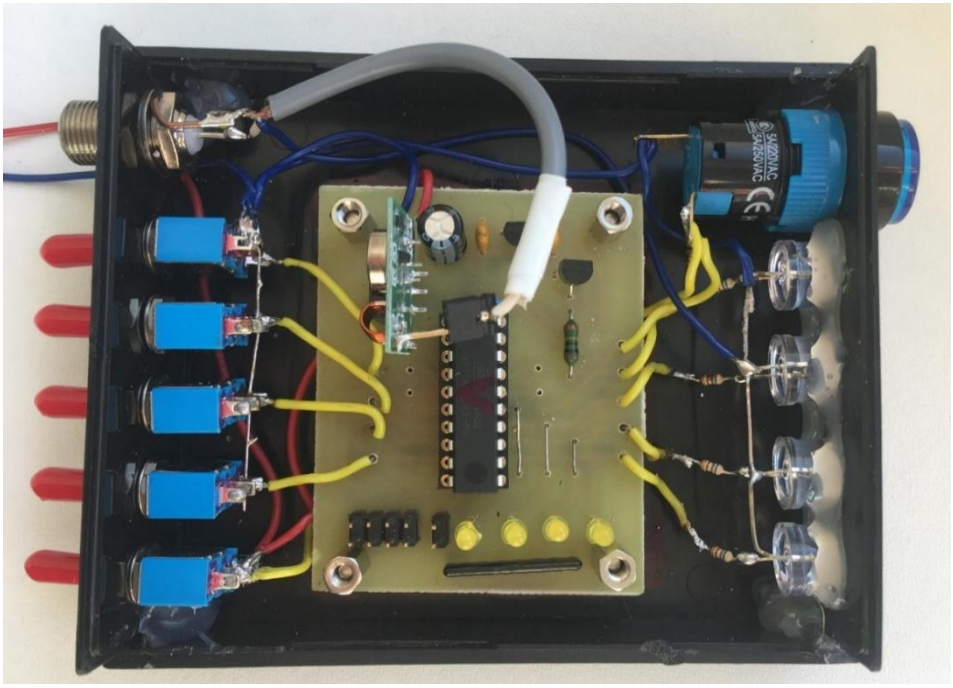
Obrázek 11

8.1.3. Seznam součástek

	Hodnota	Pouzdro	
C1	100M	C-5 mm	
C2	M1	C-5 mm	
C3	M1	C-5 mm	
D1	yellow	LED-3MM-ROUND	
D2	yellow	LED-3MM-ROUND	
D3	yellow	LED-3MM-ROUND	
D4	yellow	LED-3MM-ROUND	
D5	blue	LED-10MM-ROUND	
D6	blue	LED-10MM-ROUND	
D7	blue	LED-10MM-ROUND	
D8	blue	LED-10MM-ROUND	
D9	blue	HBS1-AY-D/B/12V	
IC1		78L05Z	TO92
IC2		DIL20S	SOCKET-20
Q1	BC546B	BC546B	TO92-EBC
R1	10K	10 mm	
R2	100K	10 mm	
R3	22K	10 mm	
R4	22K	10 mm	
R5	22K	10 mm	
R6	22K	10 mm	
RN1	1K5		SIL8
S1		P-KNX1	switch
S2		P-KNX1	switch
S3		P-KNX1	switch
S4		P-KNX1	switch
S5		P-KNX1	switch
X1		3 Pin conector	

Koaxiální konektor
Krabička plastová KM35 ABS BLACK

8.1.4. Krabička

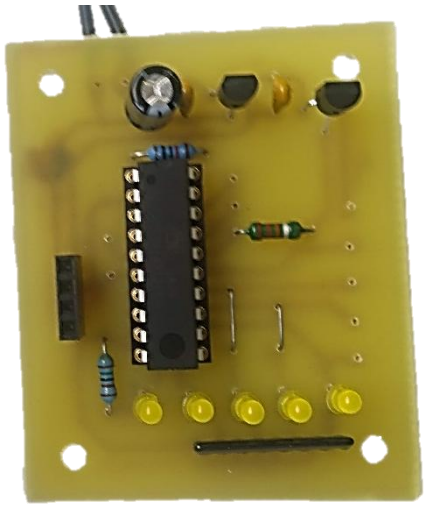


Obrázek 12

Obrázek 13

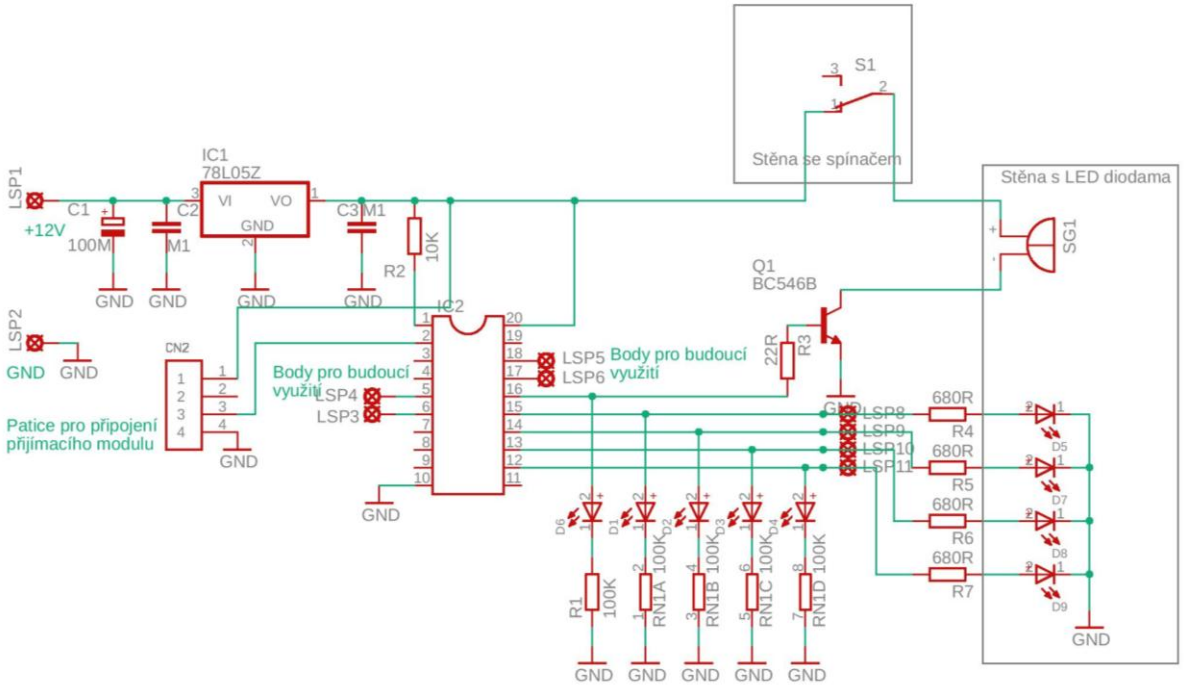
Obrázek 14

8.2. Přijímač



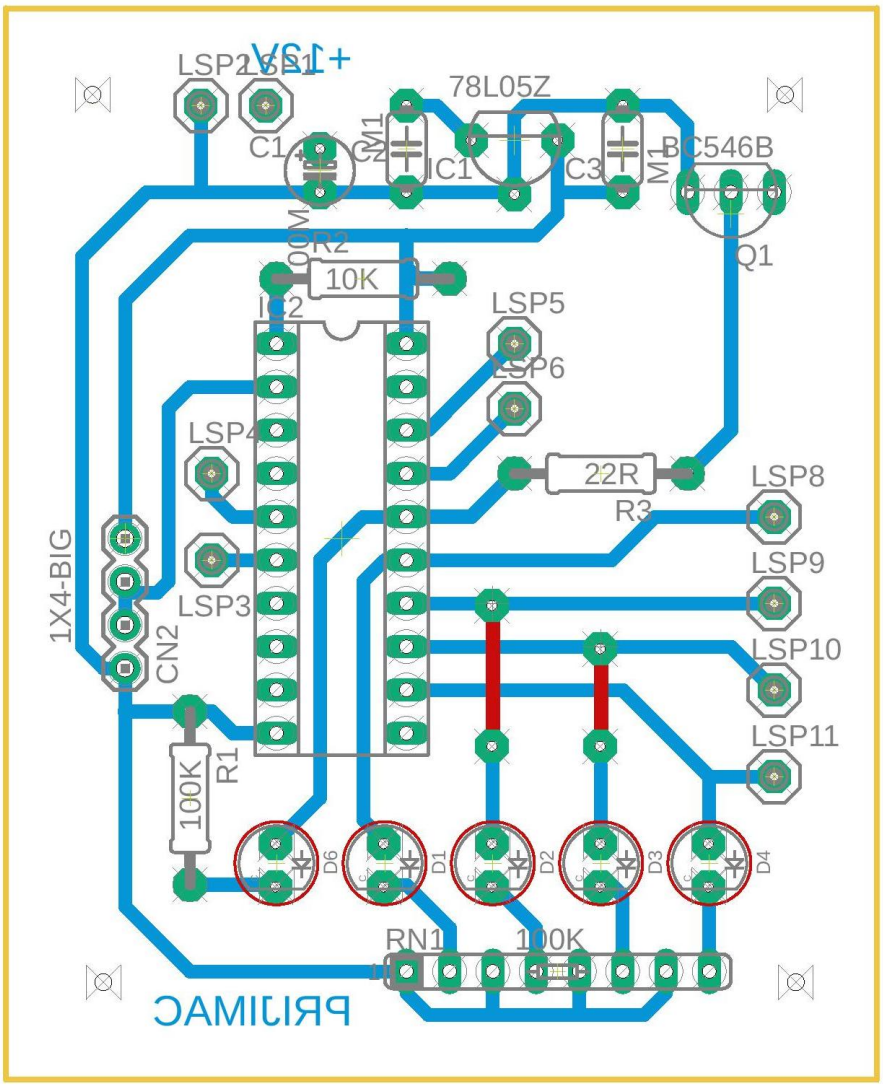
Obrázek 15

8.2.1 Schéma



Obrázek 16

8.2.2. Plošný spoj



Obrázek 17

8.2.3. Seznam součástek

	Hodnota	Pouzdro	
C1	100M	C-5 mm	
C2	M1	C-5 mm	
C3	M1	C-5 mm	
CN2		4 Pin conector	
D1	yellow	LED-3MM-ROUND	
D2	yellow	LED-3MM-ROUND	
D3	yellow	LED-3MM-ROUND	
D4	yellow	LED-3MM-ROUND	
D6	yellow	LED-3MM-ROUND	
D5	blue	LED-10MM-ROUND	
D7	blue	LED-10MM-ROUND	
D8	blue	LED-10MM-ROUND	
D9	blue	LED-10MM-ROUND	
IC1	78L05Z	78L05Z	TO92
IC2		DIL20S	SOCKET-20
Q1	BC546B	BC546B	TO92-EBC
R1	100K	10 mm rcl	
R2	10K	10 mm	
R3	22R	10 mm	
R4	680R	10 mm	
R5	680R	10 mm	
R6	680R	10 mm	
R7	680R	10 mm	
RN1	100K	SIL8	resistor-sil
S1		P-KNX1	switch
SG1			buzzer

Koaxiální konektor
Krabička plastová KM35 ABS BLACK

8.2.4. Krabička

Obrázek 18

Obrázek 19

Obrázek 20

9. Závěr

Cílem práce bylo vytvořit zařízení, které může pomoci zjednodušit identifikaci vozidel IZS ostatním řidičům a zlepšit tak jejich plynulý průjezd. Tento cíl byl dosažen, při vývoji se však projevily některé komplikace (popsané v kapitole Testování zařízení), také pro sériovou výrobu by bylo nutné provést některé úpravy.

Testováním bylo ověřeno, že zařízení je plně funkční. Modul Aurel má lepší vlastnosti než modul Arduino, což jsme předpokládali. Anténa 1 (čtvrtvlnná) vykazuje nejlepší výsledky, anténa 2 by mohla být lepší (je také podstatně menší), je však nutné ji koupit nebo vyrobit s naprostou přesností, anténa 3 je sice funkční, ale je vyrobena pro dvojnásobnou frekvenci, a proto je její dosah nejmenší. Za příznivějších podmínek bude potřeba provést ještě jeden test a ověřit předpokládaný dosah ve volném terénu (cca 200m), který nám pro daný účel připadá optimální.

Pro použití v praxi by bylo nutné použít externí krystal, aby nevznikl problém s nepřesností a kolísáním frekvence mikroprocesoru. Dále by bylo vhodné použít mikroprocesor v SMD pouzdru, protože v praxi není třeba provádět dodatečné změny programu a zařízení by bylo menší a levnější. Také by bylo vhodné celé zařízení dobře stínit a věnovat pozornost správnému připojení a instalaci antény. Pro hromadné rozšíření by bylo vhodné vyhradit vlastní frekvenci pro použití těchto zařízení, což by i eliminovalo problémy s rušením. Pro použití do automobilů by byla nutná také příslušná homologace.

Pro případné pokračování na této práci doporučujeme vyzkoušet také dosah s anténou autorádia (možnost zapojit zařízení do infoteimentu automobilu) a testovat zařízení v provozu v různém terénu (volné prostranství, les, zástavba apod.).

10. Zdroje

- Attiny4313 [online]. [cit. 2018-12-01]. Dostupné z: https://www.microchip.com/wwwproducts/en/ATtiny4313
- Gme - datasheet Attiny 4313 [online]. [cit. 2018-12-01]. Dostupné z: https://www.gme.cz/data/attachments/dsh.432-154.1.pdf
- 433 MH vysílač + přijímač. *Arduino - shop* [online]. [cit. 2019-01-18]. Dostupné z: https://arduino-shop.cz/docs/produkty/0/32/1427821401.pdf
- [online]. [cit. 2019-01-19]. Dostupné z: https://www.enika.eu/rx-bc-nbk-433-92-mhz_z1489/
- Knihovna Virtual Wire[online]. [cit. 2019-04-06]. Dostupné z: https://www.airspayce.com/mikem/arduino/VirtualWire/