CanSat pro národní a mezinárodní studentské soutěže

poznámky 4

Praha 2014 **Vladimír Váňa**

Úvod

Před několika lety se objevila zajímavá technická novinka - stavba „minisatelitů“ soukromými osobami využívajícími k jejich konstrukci plechovku od limonády. Odtud pochází i jejich název - CanSat. Poprvé s myšlenkou CanSatu přišel profesor Robert Twiggs [6] na konci 90. let. Tvůrci CanSatů do plechovky od limonády obvykle umisťují nějaká čidla jako např. čidlo tlaku a teploty, GPS moduly, kamery apod., jednočipové mikropočítače a vysílač pro VKV či UKV pásma ISP či pro některé z radioamaterských pásem 2m, 70cm, 23cm či 12cm. K vypuštění CanSatu vybaveného vlastním padáčkem obvykle slouží balon či signální raketa. Poté, co CanSat opustí raketu či balon, padá s pomocí padáčku k zemi a přitom vysílá naměřené údaje. Stavbou a provozem CanSatů se zabývají často zejména vysokoškolští studenti. Někdy je to i součást jejich studia. Příkladem může být studium SpaceMaster na ČVUT, kdy student tohoto magisterského studia je současně studentem ČVUT FEL i Luleå University of Technology, Kiruna Space Campus, Sweden a po úspěšném studiu získá tituly obou univerzit. 1. semestr absolvují studenti tohoto studia na Julius-Maximilians Universität Würzburg, Germany, kde absolvují 6 předmětů včetně XE35CSP CanSat - Projekt (3 ECTS).

Obr.1 CanSat Nederland

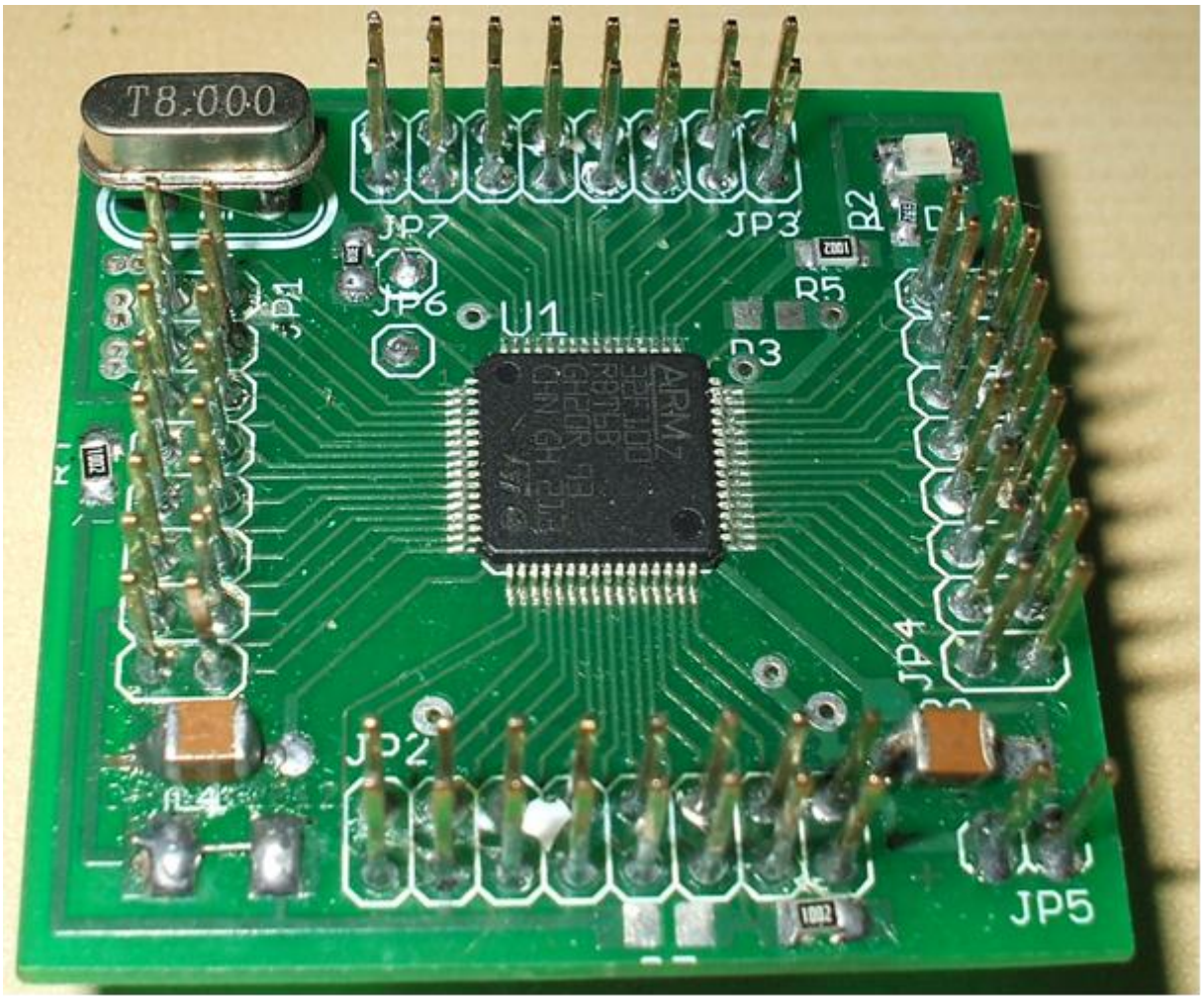
V několika posledních letech jsou dokonce pořádány národní i mezinárodní soutěže založené na použití CanSatů. V lednu 2010 vyhlásila ESA poprvé takovou soutěž i pro středoškoláky. Druhý ročník evropské soutěže proběhl v roce 2012. Evropská soutěž CanSat je součástí iniciativy ESA inspirovat mladé lidi ke studiu v oblasti vědy a inženýrství, s cílem zajistit dostupnost vysoce kvalifikovaných pracovních sil v kosmickém průmyslu budoucnosti. Finále prvních dvou ročníků soutěže se uskutečnilo na základně Andøya Rocket Range v Norsku. V roce 2010 mezi 12 finalistů postoupil i český tým X-GymZR studentů Gymnásia ze Žďáru nad Sázavou a do finále druhého ročníku 22-27.dubna 2012 tým Ječňáci studentů Střední průmyslové školy elektrotechnické z Ječné ulice v Praze [7].

Obr.2 Tým Ječňáci na základně Andøya Rocket Range v Norsku

První dva ročníky soutěže

Při prvních dvou ročnících soutěže pořádané Evropskou kosmickou agenturou soutěžily studentské týmy ze členských zemí ESA pouze v jedné kategorii. Jejich úkolem bylo vytvořit Cansat realizující tři mise a provozovat ho ve finále soutěže. První dvě mise měly všechny týmy společné – měření teploty a tlaku při sestupu Cansatu na padáčku k Zemi a přenos naměřených hodnot během sestupu bezdrátově v pásmu 433MHz do pozemní stanice týmu a jejich vyhodnocování pomocí PC. Třetí misi si již každý tým navrhl individuálně a obsah a zaměření této mise byl klíčovým pro výběru týmů do finále. Učitelé vedoucí vybrané týmy se poté účastnili workshopu v ESTEC v Holandsku. Předmětem workshopu bylo jejich seznámení s technologiemi doporučenými pro tuto soutěž v roce 2010 a 2012. Za základ splňující obě první mise byla použita stavebnice americké firmy PrattHobbies . Tuto stavebnici si po skončení workshopu také každý učitel pro svůj tým odvezl. Kromě mechanické konstrukce Cansatu obsahoval startkit i destičku vysílače, palubní počítač a čidla teploty a tlaku. Základem vysílače je integrovaný vysílač ADF7012 od Analog Devices řízený ATmega88 s firmware Stensat. Umožňuje vysílání v amatérském pásmu 70cm protokolem AX25 (packet radio). Destička palubního počítače pak je osazena ATmega 168 s firmware Arduino. To umožňuje jeho programování jazykem WIRE, což je vlastně C(++) ve kterém se neprogramuje vstupní bod main, protože ten už je v knihovně a místo něj se vytváří funkce setup (počáteční inicializace) a loop (volá se v cyklu), které se z toho knihovního main volají. Výhodou je i značné množství knihovních funkcí použitelných ve firmware Cansatu (GPS, čidla teploty, tlaku, akcelerometry atd.). Přijímač pozemní stanice, anténu a počítač si soutěžní týmy již musely zajistit samy. V případě týmu Ječňáci jsme získali grant HMP. Na výsledném hodnocení se technická stránka řešení podílí 50 %. Kromě technického řešení a dokumentace jsou dále hodnoceny měsíční reporty zasílané týmy do ESA, jejich publikační a propagační činnost, www stránky, blog a rovněž prezentace před a po letu jejich Cansatu. Veškerá komunikace s ESA, reporty, presentace apod. jsou pochopitelně pouze v angličtině.

Při druhém ročníku soutěže se technické řešení českého týmu dosti lišilo od řešení ostatních studentských týmů. Při evropské soutěži jsme nechtěli za základ vzít americkou stavebnici s americkými obvody a navrhli a realizovali jsme elektroniku vlastní, založenou na obvodech největšího evropského výrobce mikroelektronických prvků italsko-francouzské firmy STMicroelectronic. Tato firma má v Praze 8 i své významné vývojové centrum. Palubní počítač jsme osadili 32bitovým MCU ARM Cortex STM32F100.



Obr.3 Palubni počítač s STM32F100

Rovněž čidla jsme použili od STM. Protože soutěž má studentům přiblížit skutečný kosmický výzkum měl náš tým několik Cansatů (technologický a dva letové). Další, čím se naše řešení lišilo od řešení ostatních týmů bylo použití trojice padáčků čímž se zvýšila stabilita klesajícího Cansatu, což bylo kladně hodnoceno pořadatelem i ostatními týmy. Bohužel vyhazovací zařízení rakety (systém pružin) prorazilo plech plechovky našeho Cansatu a úlomek tohoto plechu přerušil tištěný spoj vysílače, takže vysílač stačil odeslat jen několik paketů. Z časových důvodů nedošlo ke startu další rakety s našim záložním letovým kusem Cansatu. Podmínky soutěže předepisují rozměry Cansatu stejné, jako má plechovka 350 ml, ale není zde podmínka tuto plechovku použít. Většina týmů sice plechovku coby kryt mechanické konstrukce používala, nicméně několik týmů použilo vlastní bytelnější konstrukci.

Třetí ročník soutěže

Třetí ročník soutěže se od předchozích poněkud lišil. Na základě předchozích zkušeností totiž ESA vytvořila dvě soutěžní kategorie beginners a advanced. Takže z jedné země mohou postoupit i dva týmy, Na druhé straně některé země pak nejsou zastoupeny vůbec. Z České republiky bohužel nepostoupil žádný tým (tým Ječňáci se do třetího ročmíku soutěže již nepřihlásil, protože všichni jeho členové v tomto roce maturovali). Nejbližší účastníci letošní soutěže jsou v kategorii začátečníků tým Kraksat studentů z Krakova a v kategorii pokročilých tým Aurora studentů z Skierniewice rovněž z Polska . Další změnou je to, že na učitelském workshopu dostali vedoucí týmů konečně „evropský“ startkit. Byl vytvořen na dánské univerzitě Aalborg Univerzity . Jako palubní počítač používá startkit Arduino Uno s ATmega328 americké firmy Atmel. Komunikaci v pásmu 433MHz zajišťuje dvojice trancieverů APC220 výrobce např. Shenzhen Shanhai Technology Ltd osazených integrovanými trancievery americké firmy Analog Devices ADF7020.

Čtvrtý ročník soutěže

Čtvrtý ročním soutěže byl vyhlášen 7.října 2013. Do finále v kategorii *advanced* postoupil do finále tým *Pragsat* studentů Střední průmyslové školy elektrotechnické v Ječné ul. V Praze. To je také důvodem vzniku předkládaného materiálů, neboť jeho úkolem je podpora studentů při vývoji jejich minisatelitu. Za základ byl vzat startkit vyráběný od ledna 2014 firmou T-minus právě pro evropské soutěže ESA. Tento startkit jsme získali na *Teachers' introductory workshop at ESA/ESTEC* 24. a 25.ledna 2014. Bohužel na workshopu byli učitelé seznámeni s převážně technologiemi používanými v předchozím ročníku. O startkitu od T-minus jsme získali jen minimum informací. Důvodem je možná i to, že jde o nový výrobek malé firmy. Proto jsem moduly získaného startkitu (palubní počítač s ATMega a tranciever pro 433 MHz) podrobil zkoumání a výsledky publikoval, spolu s vlastními moduly v tomto materiálu. Evropské finále soutěže, jahož se účastnil i náš tým Pragsat se konalo 1 - 5.června 2014 na základně Andøya Rocket Range v Norsku.

Obr.4 Tým PragSAT na základně Andøya Rocket Range v Norsku

Protože tato skripta navazují na skripta z předchozích let [1],[2],[3], jsou zaměřena na podporu vývoje cansatu pro ESA soutěž 2014 a neopakují informace z předchozích skript.

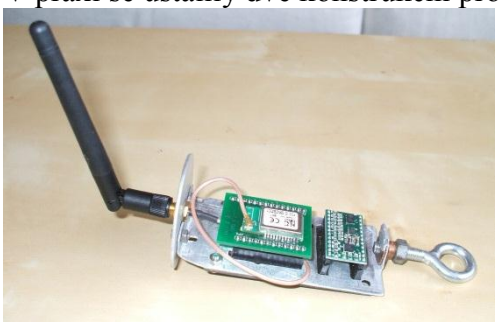
1. Architektura CanSATu

1.1Hardware

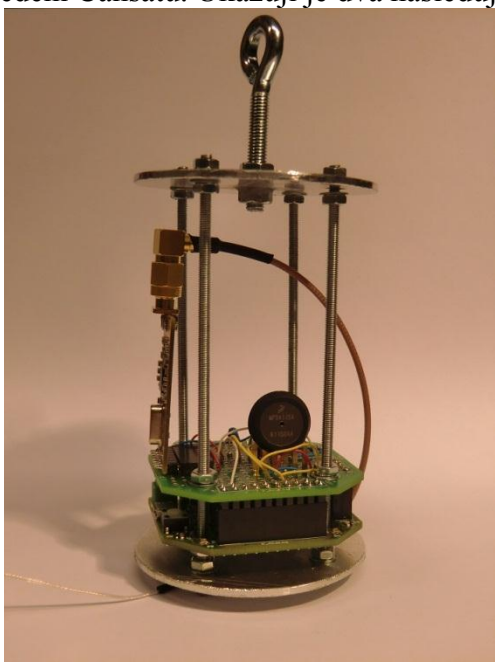
Elektronické vybavení Cansatu je dáno jeho požadovanou funkcí. Bude obsahovat blok čidel teploty, atmosférického tlaku, 3D akcelerometr a 3D magnetometr. Dále bude obsahovat palubní počítač pro komunikaci s čidly (nastavení jejich vlastností, přečtení dat) a z vysílače. Dalším blokem tedy bude ještě vysílač. Protože v případě poruchy radiového spojení bychom přišli o naměřená data, doplnili jsme elektroniku ještě o záznam dat na SD kartu. Nedílnou částí elektroniky je i napájecí zdroj.

1.2 Konstrukční provedení

V praxi se ustálily dvě konstrukční provedení Cansatů. Ukazují je dva následující obrázky.

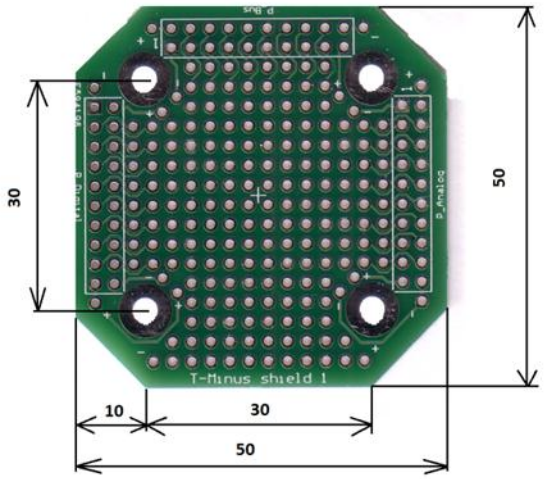


Obr.5



Obr.6

Zvolili jsme variantu z obr.6, protože máme dvojici trancieverů 433MHz vyráběné holandskou firmou T-minus pro ESA. Tím jsou ale určeny rozměry desek plošných spojů námi vyvíjených bloků.



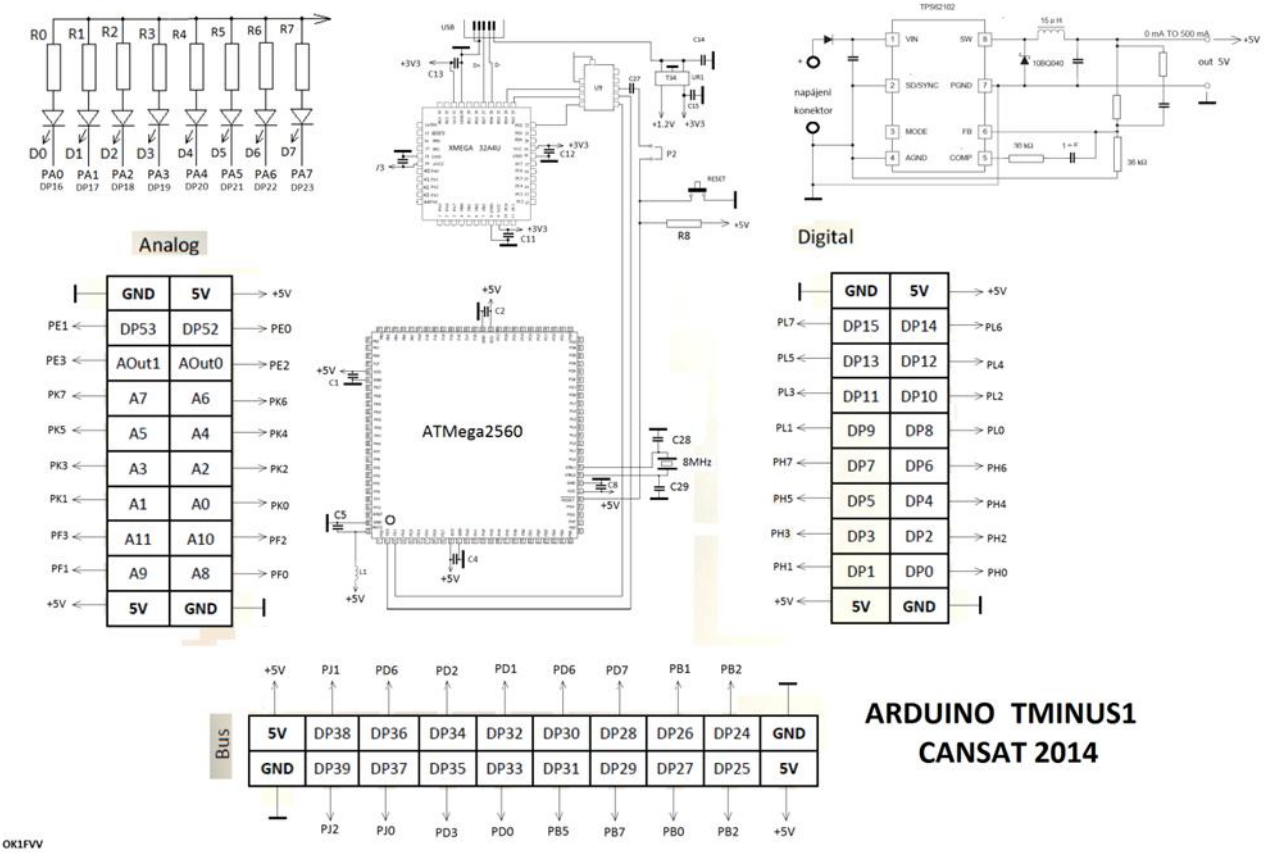
Obr.7

Čtyři otvory o Ø 3,2 mm slouží k navlečení desek na čtveřici svorníků se závitem M3 – viz obr.7

2. Palubní počítač

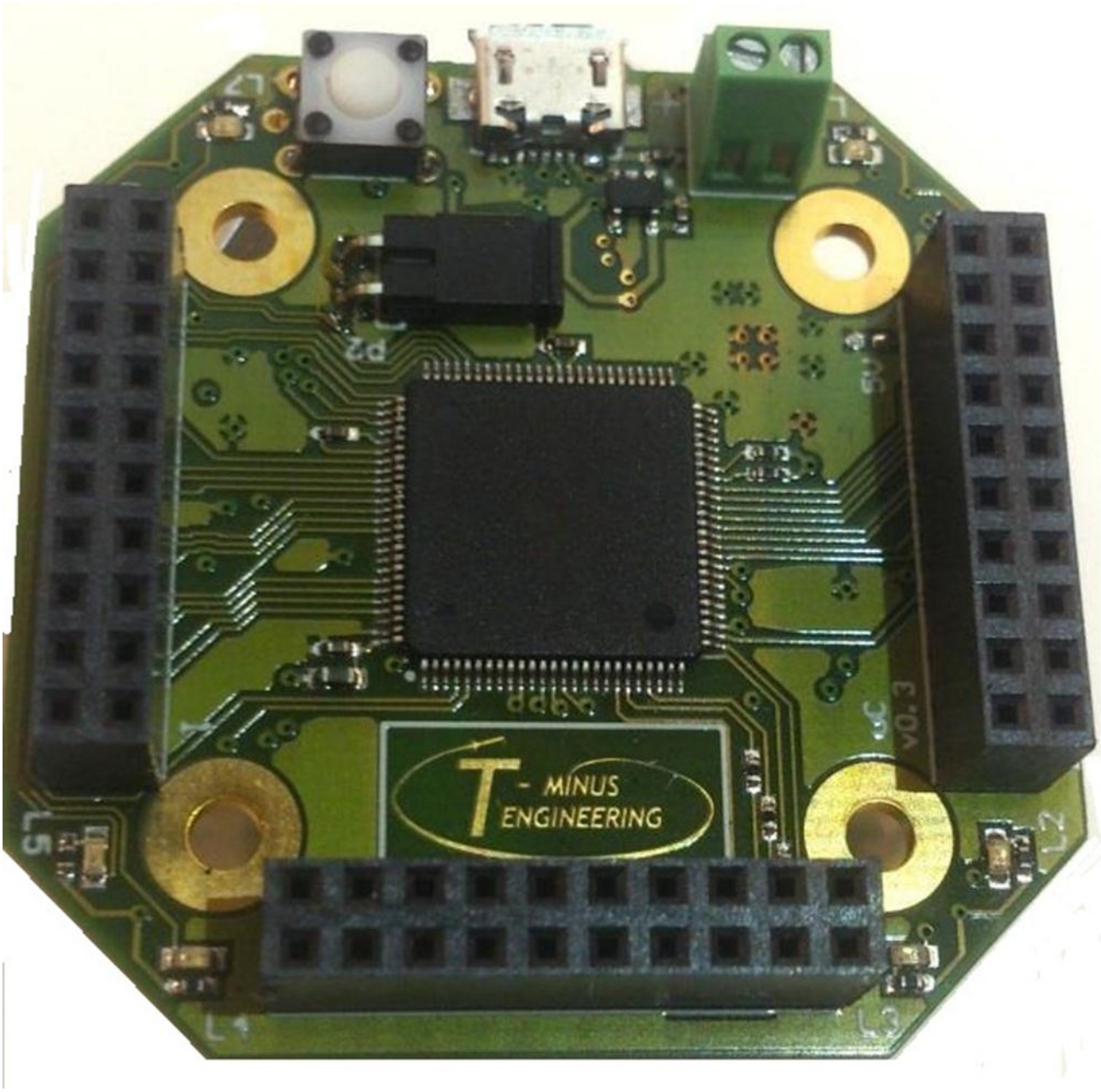
2.1 Palubní počítač od T-minus

V tomto materiálu popisuji své poznatky týkající se flight computeru od T-minus . Tento počítač je součástí startkitu, který jsme získali na workshopu v ESA ESTEC. Bohužel převážná většina informací na workshopu se týkala loňské stavebnice s Arduino UNO a tcvr APC220 a nikoli startkitu T-minus.

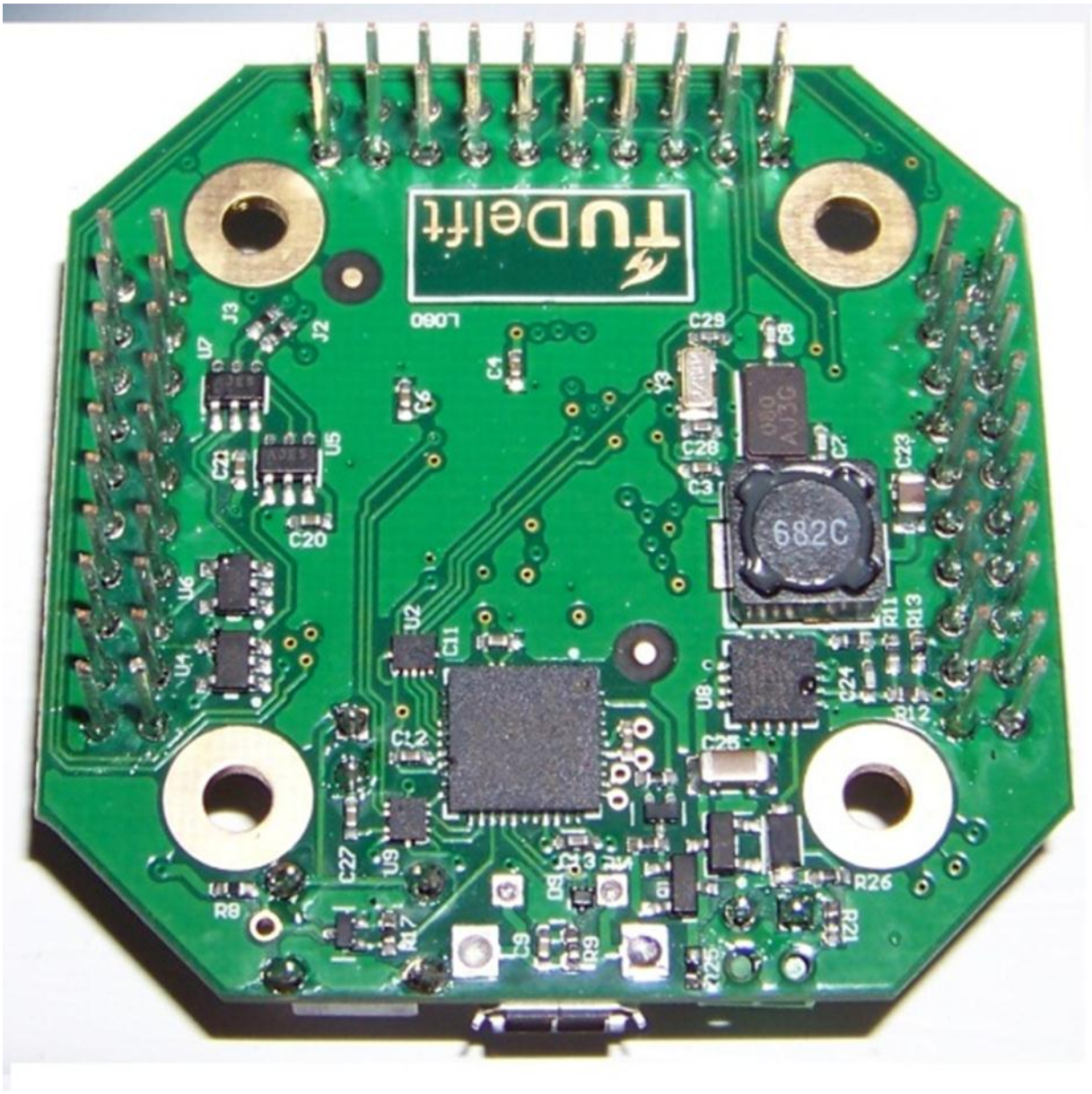


Obr.8 zapojení palubního počítače T-minus

Na horní straně pcb je umístěna ATMega2560 coby Arduino Tminus1. Netváří se tedy jako např. Arduino Mega2560 či Arduino ADK . Na rozdíl od nich nemá hodinový kmitočet 16MHz, ale jen 8MHz. Má s nimi však společné to, že ATMega 2560 je napájena 5V a tedy pracuje na 5V logice. Napájení 5V pro ATMega2560 se získá buď z PC přes USB, nebo z vnějšího zdroje připojeného k dvojici svorek vedle usb konektoru. Napájecí napětí může být cca 3V až 9V. Za + svorkou najdeme ještě ochrannou diodou a pak již Multimode Low-Power Buck Konvertor TPS62100 od Texas Instruments v doporučeném zapojení, na jehož výstupu je již potřebných 5 V. Převážná většina pinů ATMega2560 je připojena coby digitální či analogové porty arduina na trojici konektorů. Kromě toho je dalších 8 digitálních pinů použito pro 8 LED. Ty jsou po dvou umístěny v každém rohu horní strany pcb.
Na rozdíl od většiny současných Arduino, do nichž se sketche z PC přes USB nahrávají prostřednictvím interface s ATMega16A4U je zde použita pinově kompatibilní, ale nízkopříkonová XMega 32A4U. Ta je na rozdíl od ATMega2560 napájena 3V3. Ty poskytuje obvod UR1 napájený 5V z konektoru USB. Stejně jako u jiných arduino jsou PD2 a PD3 XMega32U4 spojeny s PE0 a PE1 ATMega2560 a PD4 XMega32U4 přes kondenzátor C27 (M1) a propojku P2 s RESET/ ATMega2560. Vzhledem k rozdílným napájecím napětím XTMega32U4 a ATMega2560 a tedy i logickým úrovním však tato trojice signálů prochází obvodem U9, sloužícím k převodu úrovní.

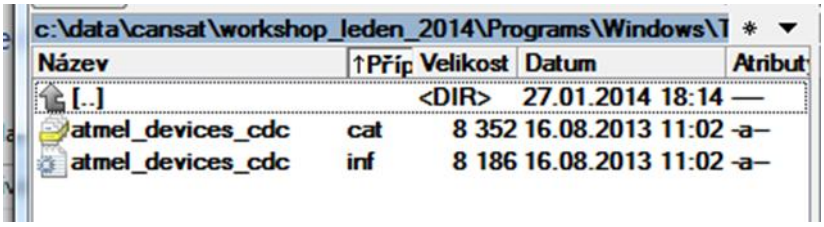


Obr.9 palubní počítačT-minus horní strana

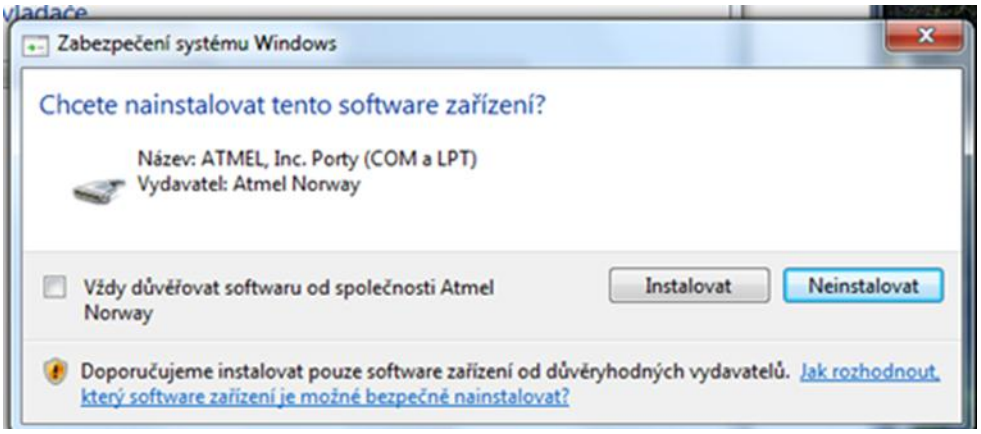


Obr.10 palubní počítač T-minus dolní strana

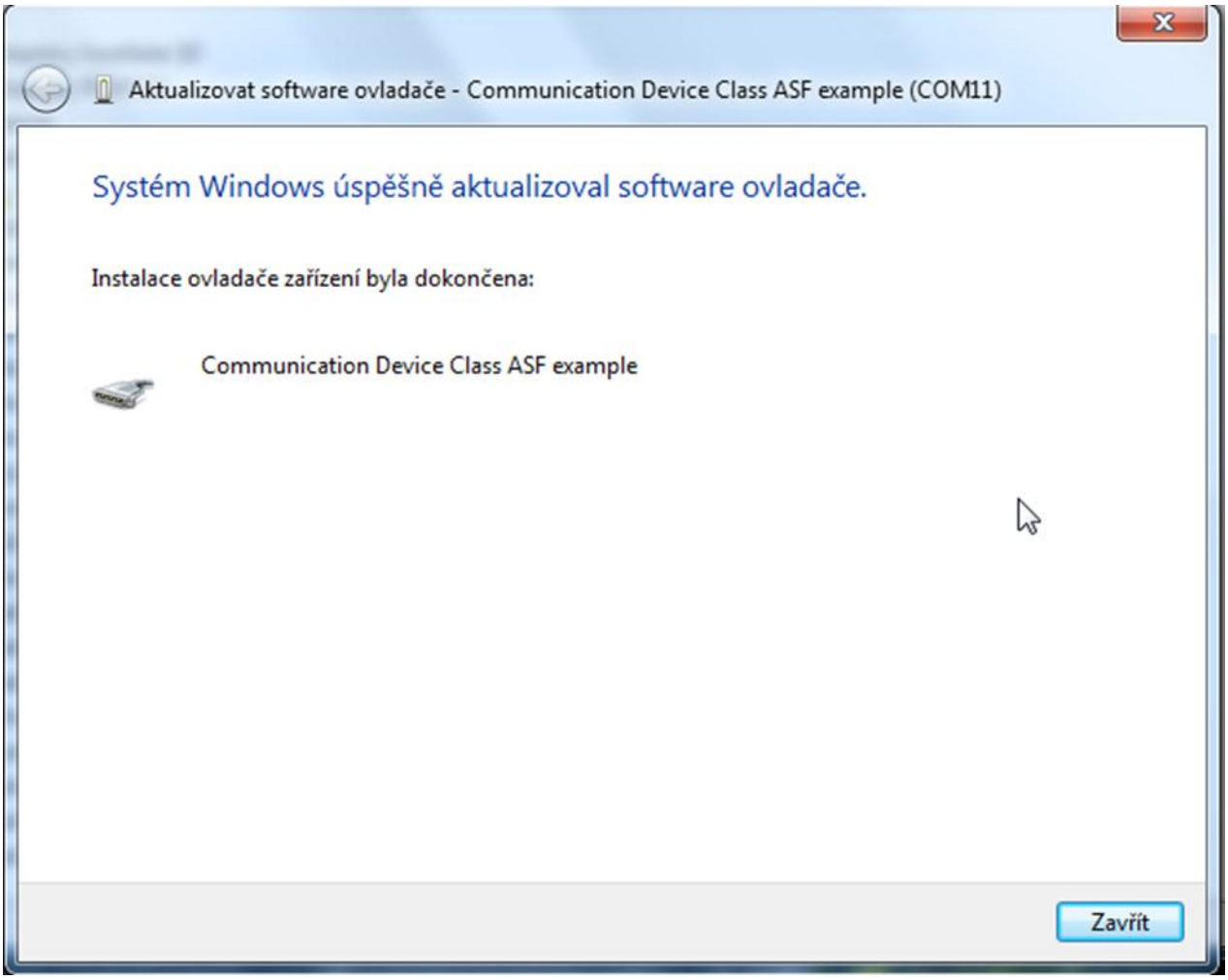
Dále si popíšeme práci s tímto palubním počítačem.
Pokud poprvé připojíme desku Flight computeru přes usb k PC (např. s WIN7), OS zjistí nové usb zařízení a pokusí se (neúspěšně) nainstalovat drivery. Proto doinstalujeme potřebné drivery ručně.



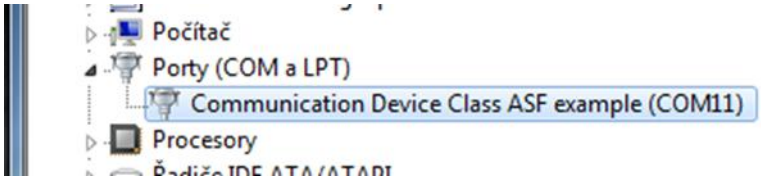
Tento ovladač jsme získali na workshopu v ESA ESTAC a jmenuje se atmel_devices_cdc (stejný ovladač budeme používat i tranciever na 433MHz). Tento driver nainstalujeme:



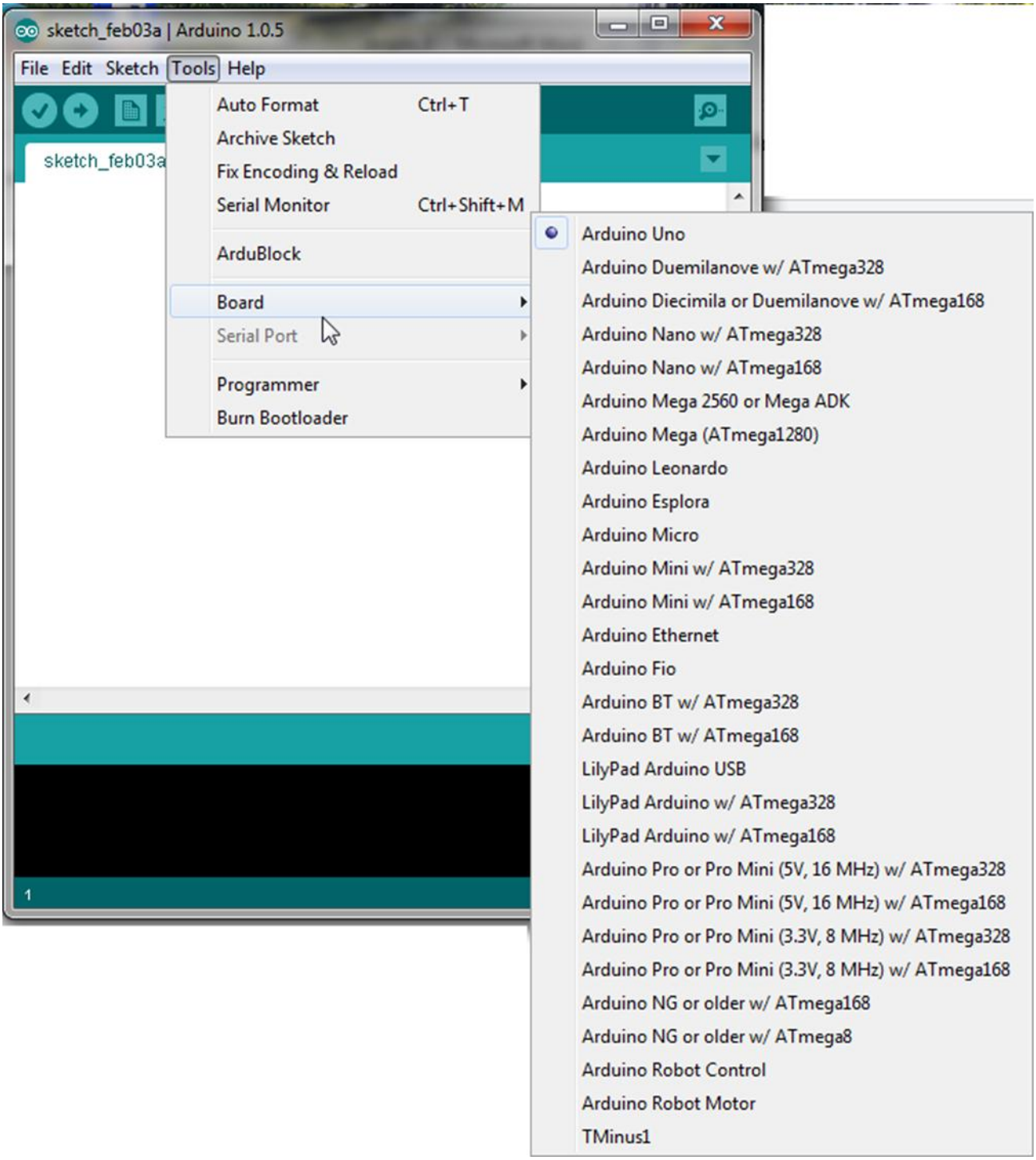
Klikneme na Instalovat.



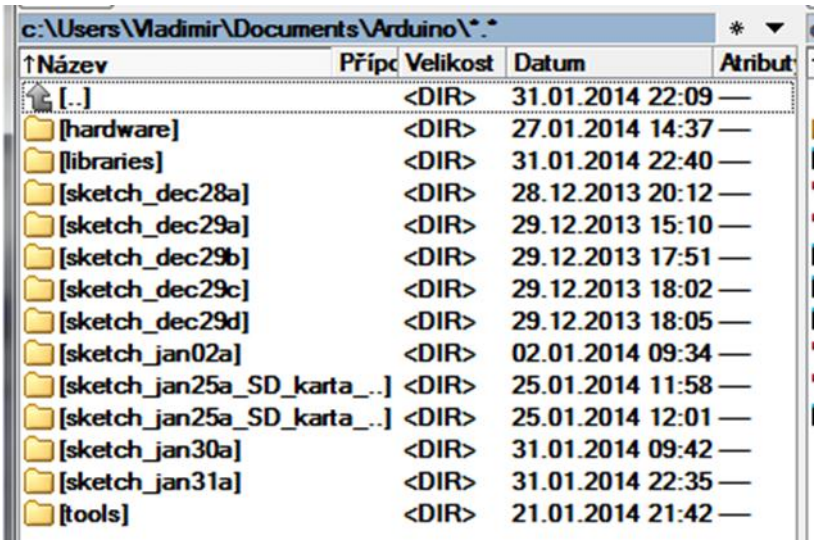
Driver je úspěšně nainstalován. Ve správci zařízení se objeví virtuální port se zařízením **Communication Device Class ASF**



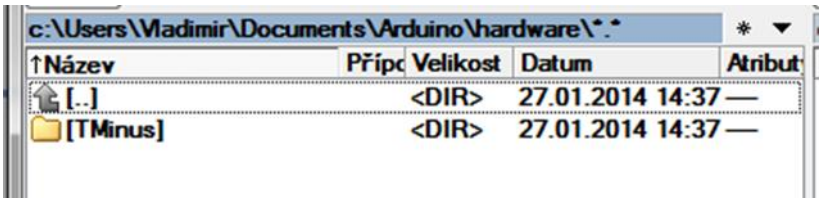
K spolupráci *flight computeru* a **IDE Arduina** ještě potřebujeme IDE doplnit o položku TMinus1 (viz menu Tools → Boards)



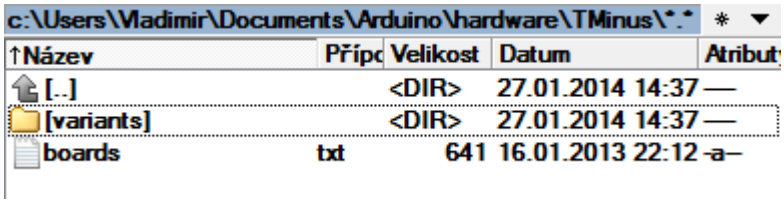
Provedeme to jednoduše tak, že vytvoříme složku hardware v adresáři, který používá IDE Arduina



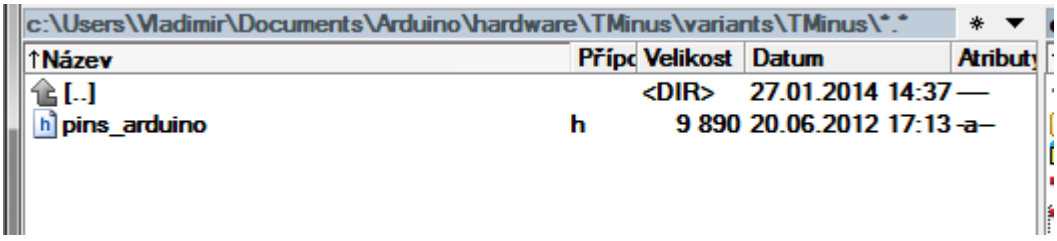
A do tohoto adresáře zkopírujeme složku TMinus



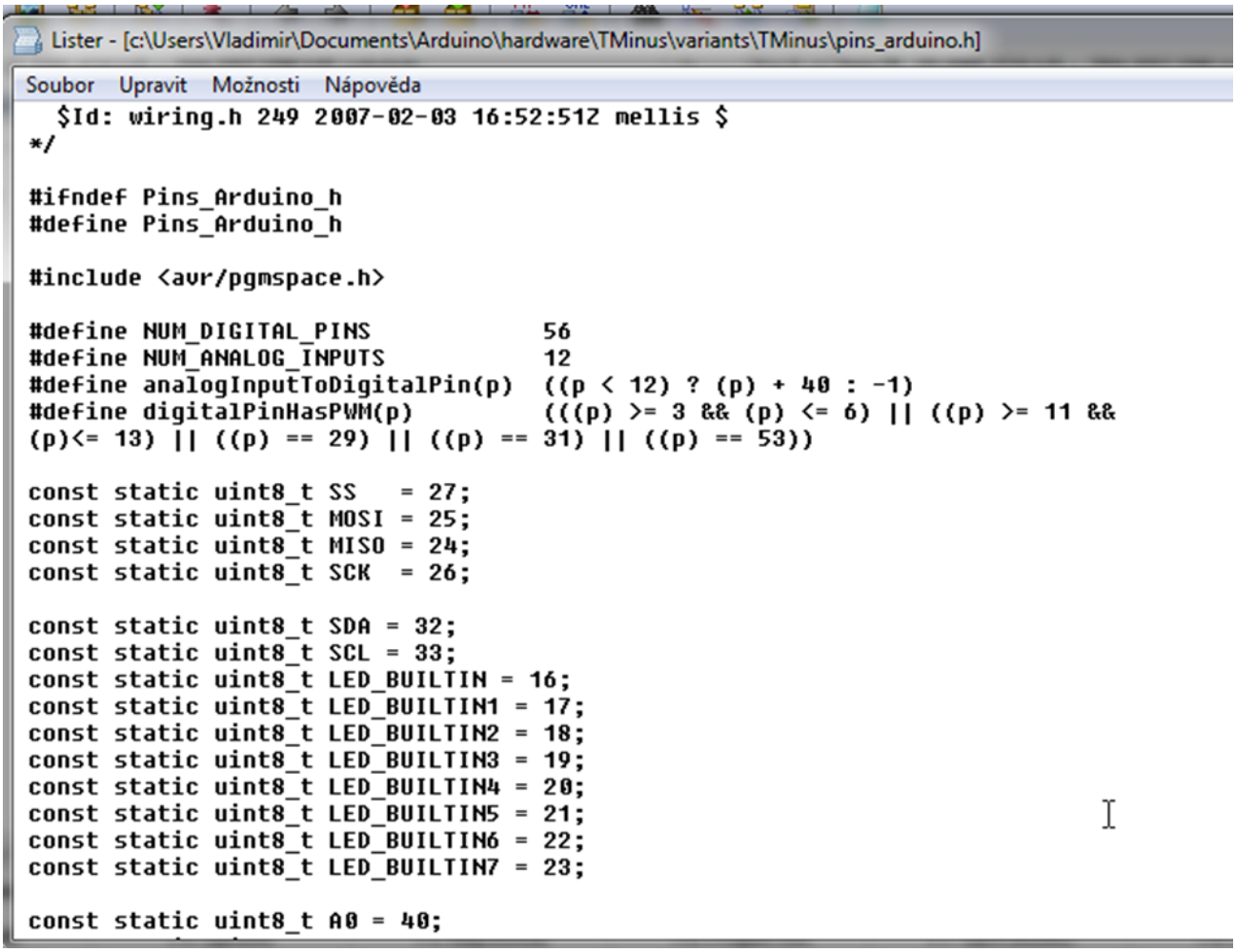
Se soubory boards.txt



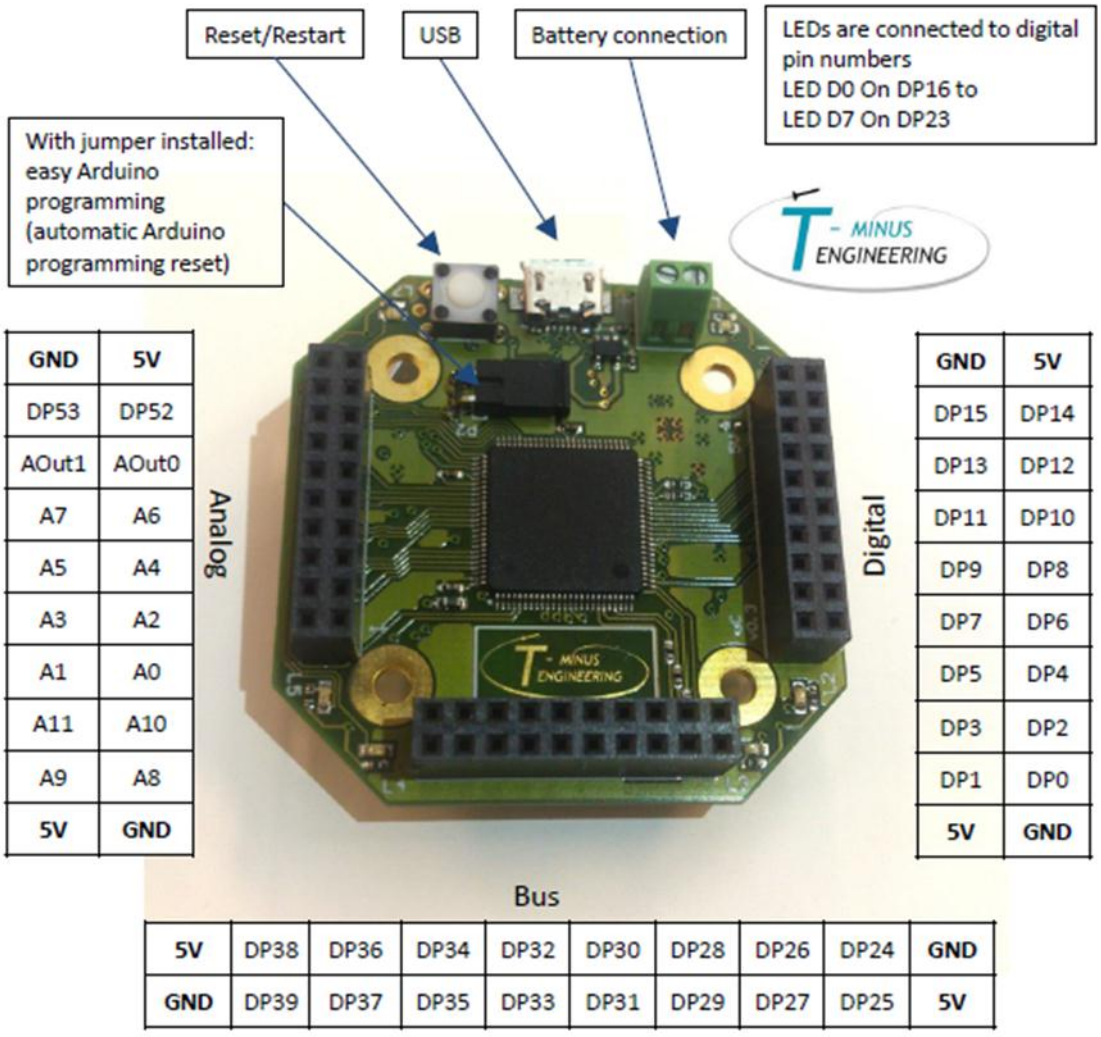
a pins_arduino.h



Tento soubor jsme získali v dokumentaci na workshopu v ESA ESTEC.



Pro další práci s palubním počítačem ještě potřebujeme rozložení signálů na pinech konektorů.



GND	5V
DP53	DP52
AOut1	AOut0
A7	A6
A5	A4
A3	A2
A1	A0
A11	A10
A9	A8
5V	GND

GND	5V
DP15	DP14
DP13	DP12
DP11	DP10
DP9	DP8
DP7	DP6
DP5	DP4
DP3	DP2
DP1	DP0
5V	GND

5V	DP38	DP36	DP34	DP32	DP30	DP28	DP26	DP24	GND
GND	DP39	DP37	DP35	DP33	DP31	DP29	DP27	DP25	5V

Pin description

Arduino digital pin numbers	pin on port	connected to	function	Port	Bit of port	timer	Interrupt (IN)
0	PH0	Digital	digital I/O	PH	0		
1	PH1	Digital	digital I/O	PH	1		
2	PH2	Digital	digital I/O	PH	2		
3	PH3	Digital	digital I/O/ PWM	PH	3	OCR4A	
4	PH4	Digital	digital I/O/ PWM	PH	4	OCR4B	
5	PH5	Digital	digital I/O/ PWM	PH	5	OCR4C	
6	PH6	Digital	digital I/O/ PWM	PH	6	OCR2B	
7	PH7	Digital	digital I/O	PH	7	T4	
8	PL0	Digital	digital I/O	PL	0		
9	PL1	Digital	digital I/O	PL	1		
10	PL2	Digital	digital I/O	PL	2	T5	
11	PL3	Digital	digital I/O/ PWM	PL	3	OCR5A	
12	PL4	Digital	digital I/O/ PWM	PL	4	OCR5B	
13	PL5	Digital	digital I/O/ PWM	PL	5	OCR5C	
14	PL6	Digital	digital I/O	PL	6		
15	PL7	Digital	digital I/O	PL	7		
16	PA0	Onboard LED		PA	0		
17	PA1	Onboard LED		PA	1		
18	PA2	Onboard LED		PA	2		
19	PA3	Onboard LED		PA	3		
20	PA4	Onboard LED		PA	4		
21	PA5	Onboard LED		PA	5		
22	PA6	Onboard LED		PA	6		
23	PA7	Onboard LED		PA	7		
24	PB3	BUS	MISO	PB	3		
25	PB2	BUS	MOSI	PB	2		

26	PB1	BUS	SCK	PB	1		
27	PB0	BUS	/SS	PB	0		
28	PD7	BUS	T0	PD	7	T0	
29	PB7	BUS	OC0A	PB	7	OCR0A OCR1C	
30	PD6	BUS	T1	PD	6	T1	
31	PB5	BUS	PWM	PB	5	OCR1A	
32	PD1	BUS	SDA	PD	1		1(use3)
33	PD0	BUS	SCL	PD	0		0(use2)
34	PD2	BUS	RXD1	PD	2		2(use4)
35	PD3	BUS	TXD2	PD	3		3(use5)
36	PD5	BUS	XCK1	PD	5		
37	PJ0	BUS	RXD3	PJ	0		
38	PJ1	BUS	TXD3	PJ	1		
39	PJ2	BUS	XCK3	PJ	2		
40	PF0	Analog	ADC0 (A8)	PF	0		
41	PF1	Analog	ADC1 (A9)	PF	1		
42	PF2	Analog	ADC2 (A10)	PF	2		
43	PF3	Analog	ADC3 (A11)	PF	3		
44	PK0	Analog	ADC8 (A0)	PK	0		
45	PK1	Analog	ADC9 (A1)	PK	1		
46	PK2	Analog	ADC10 (A2)	PK	2		
47	PK3	Analog	ADC11 (A3)	PK	3		
48	PK4	Analog	ADC12 (A4)	PK	4		
49	PK5	Analog	ADC13 (A5)	PK	5		
50	PK6	Analog	ADC14 (A6)	PK	6		
51	PK7	Analog	ADC15 (A7)	PK	7		
52	PE2	Analog	AIN0	PE	2		
53	PE3	Analog	PWM/AIN1	PE	3	OCR3A	
54	PE0	USB	TXD0	PE	0		
55	PE1	USB	RXD0	PE	1		

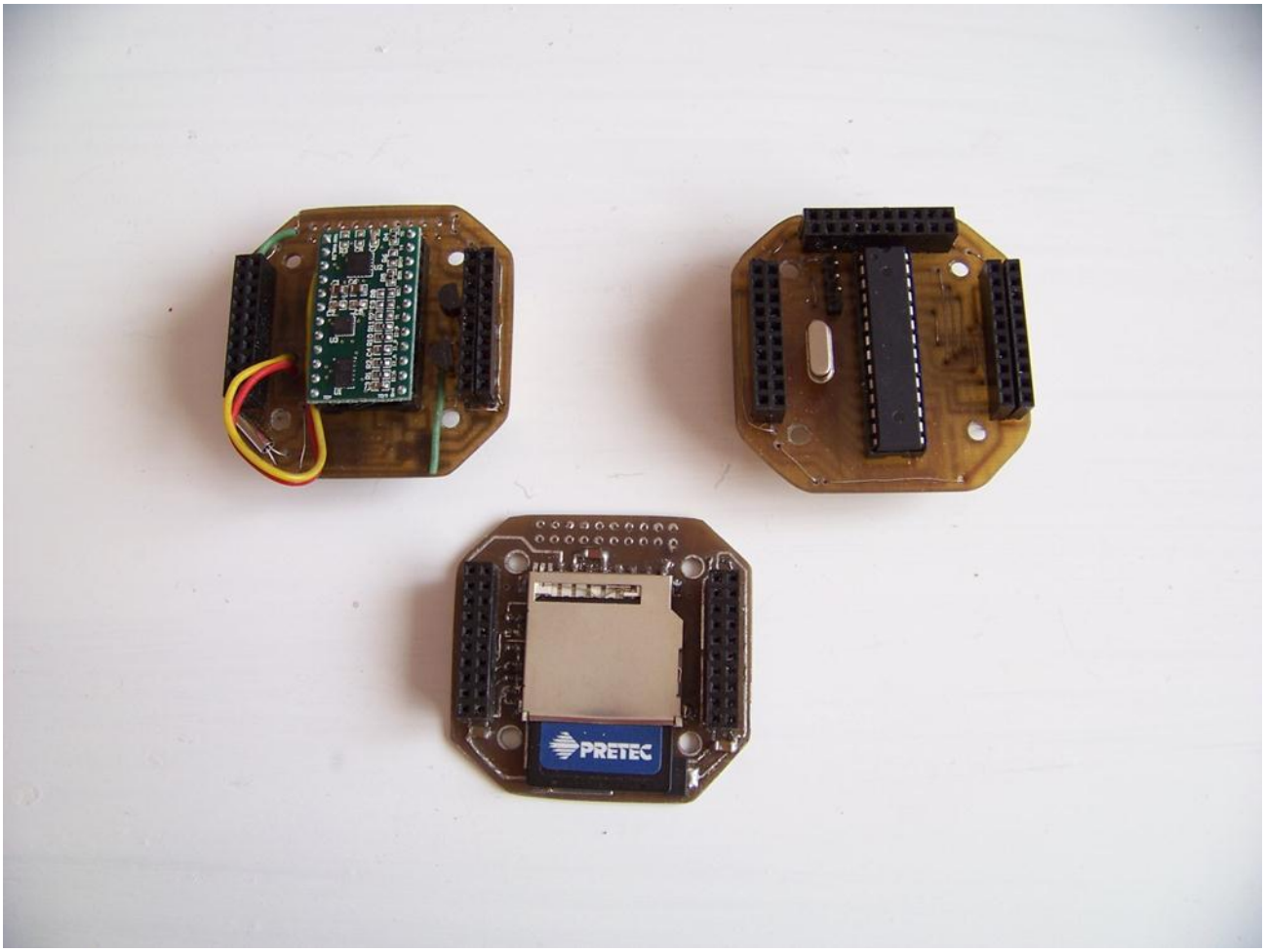
2.2 Vlastní palubní počítač s ATMega328

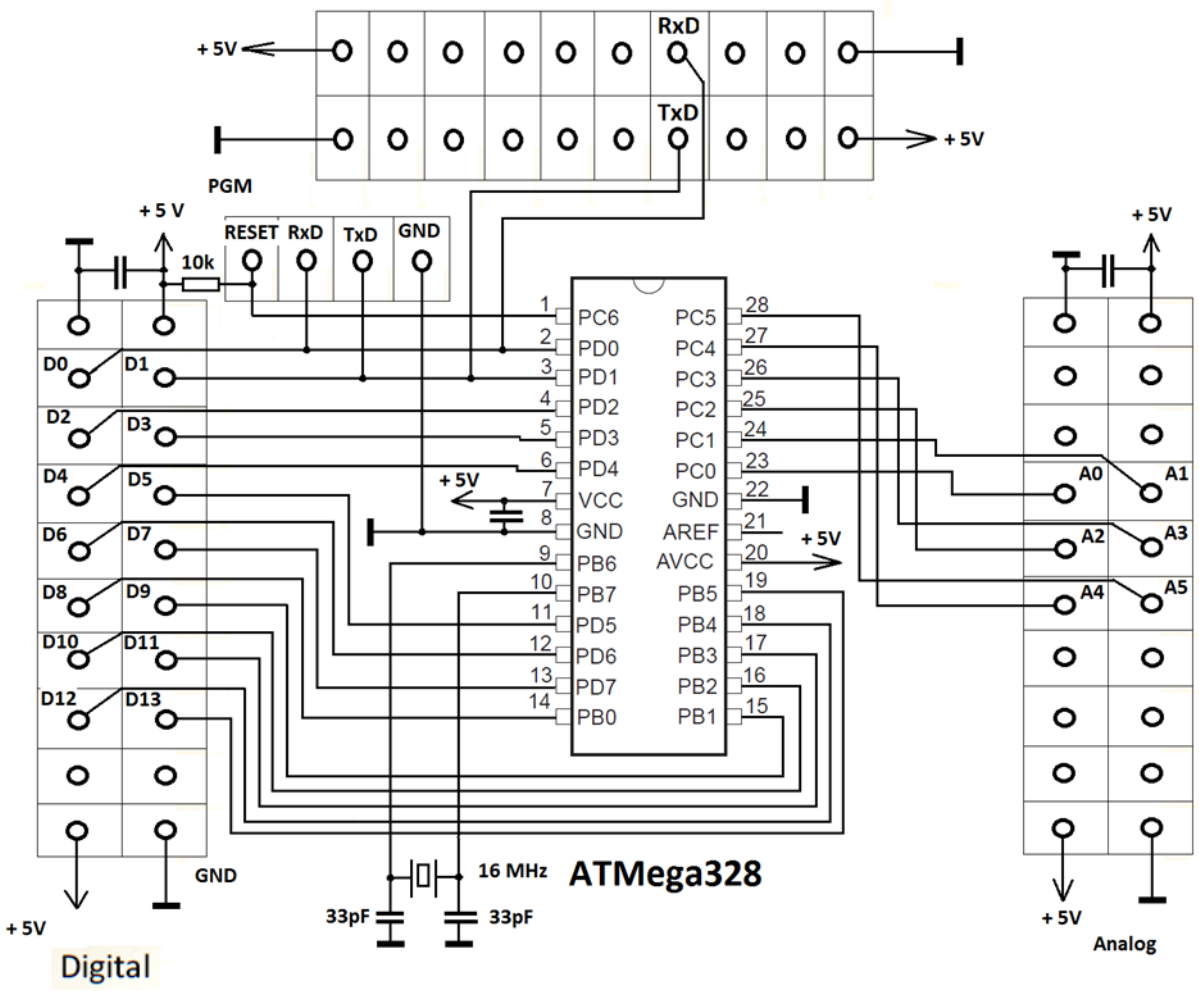
Stavebnice CanSATu přivezená z Workshopu v ESA ESTEC (výroba T-minus) obsahuje destičku palubního počítače s ATMega2560 a systémem Arduino a dále 2 destičky trancieveru 433 MHz s TRC105 firmy RFM. Kromě toho též jednu univerzální destičku pro sensory apod. Největší výhodou tohoto startkitu je, že jsme zadarmo dostali téměř celou hotovou elektroniku. Pokud bychom na univerzální destičku umístili destičku s čidly od STMicroelectronics STEVAL MKI124V1 a dále jeden IO pro převod napájení 5V napájení ze startkitu na 3V3 pro čidla (např.LM3940IMP-3.3/NOPB od Texas Instruments) a ještě 2 tranzistory řízené polem (např.BSS138 nebo BS170) pro převod logických úrovní signálů SDA a SCL komunikace I2C, máme minimální verzi hw hotovou a stačí dopsat sw, což pro Arduino je jednoduché. Drobnou nevýhodou je, že máme jen jedinou elektroniku pro Cansat a jakékoliv poškození, ztráta apod. může mít fatální následky.

U reálných kosmických programů jsou fatální následky mnohem větší, než u studentského projektu. Šlo by o finanční ztráty v řádech milionů dolarů a další ztráty mnohdy nevyčíslitelné. Proto se vždy zhotovuje celá řada ekvivalentních či téměř ekvivalentních zařízení, satelitů apod. Jde např. o technologický (technologická) zařízení a několik kusů zařízení letových.

Protože cílem soutěží ESA EDUCATION je připravovat studenty pro práci na reálných kosmických projektech pokusíme se i my zhotovit několik kusů CanSATů. Na popisované minimální verzi CanSATu jsem ověřil, že vyvinout a vyrobit obdobný minisatelit může i jeden člověk v poměrně krátké době několika dní či týdnů nepřesahující 1 měsíc. Je tak i

možné, aby několik členů týmů vyrobilo své různé konstrukce a do soutěže se pak vybrala ta nejlepší, popř. Cansat složený z nejlepších prvků jednotlivých cansatů. Nevýhodou může být, že by se tím mohla trochu omezit diskuze nutná při zhotovování řešení jediného.
Minimální verze je složena ze tří destiček – palubního počítače s ATMega328 s Arduino bootloaderem, destičky s čidly a destičky s SD kartou. Jako tranciever lze použít jak tcvr od T-minus, tak tcvr APC220, se kterým se soutěžilo v roce 2013, popř. nějakým vlastním tcvr.

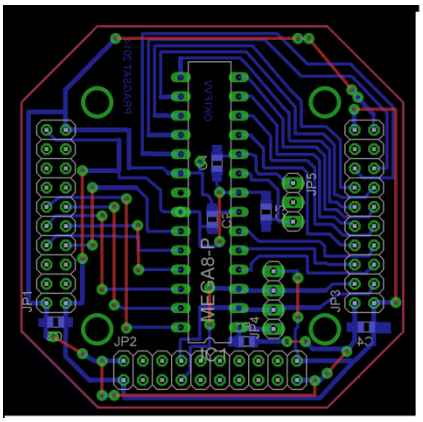


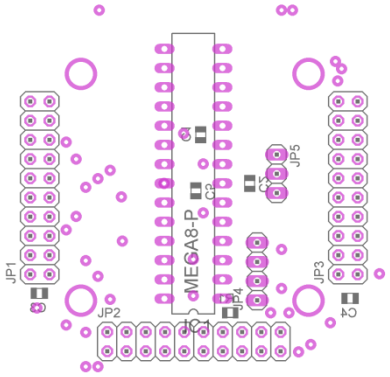


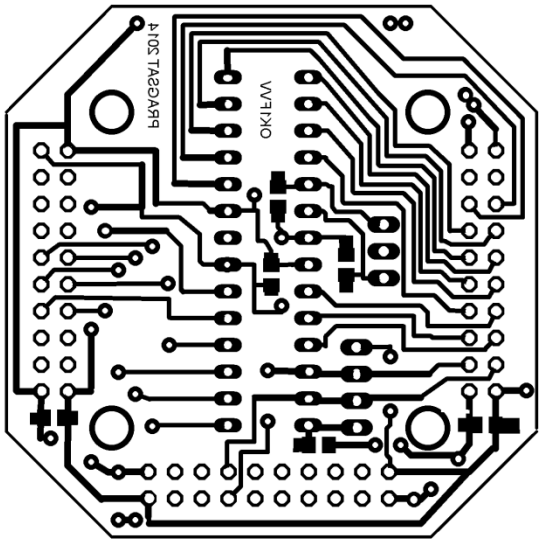
Palubní počítač compatib. s ARDUINO UNO

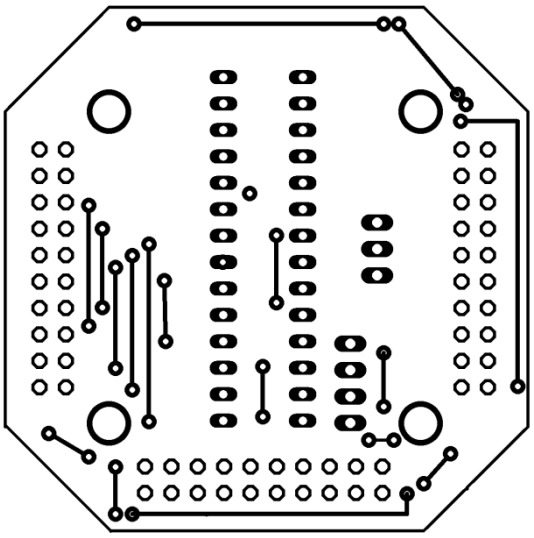
OK1FVV

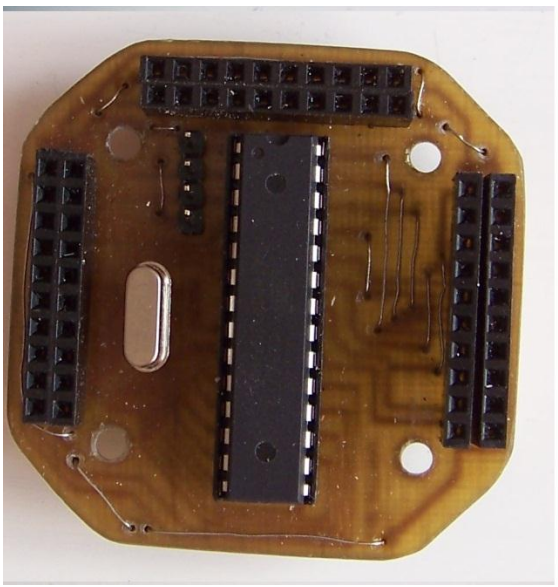
Zapojení počítače je velice jednoduché – obsahuje krystal 16MHz stejně jako ARDUINO UNO a dále obsahuje již jen propojení svých pinů s odpovídajícími dutinkami konektoru tak, aby tato deska byla v rozsahu D0 –D13 a A0 až A5 kompatibilní s počítačem od T-minus. Potom obsahuje již jen 4 pinový konektor pro připojení programátoru Arduino, např. s FT232RL ze stavebnice Cansatu z 2012. Dále uvádím PCB:

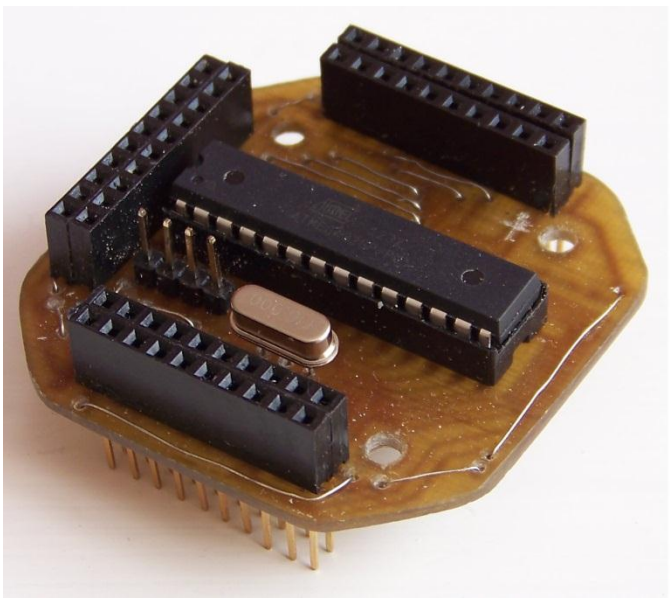

MEGA8-P
JP1
JP2
JP3
JP4
JP5


MEGA8-P
JP1
JP2
JP3
JP4
JP5

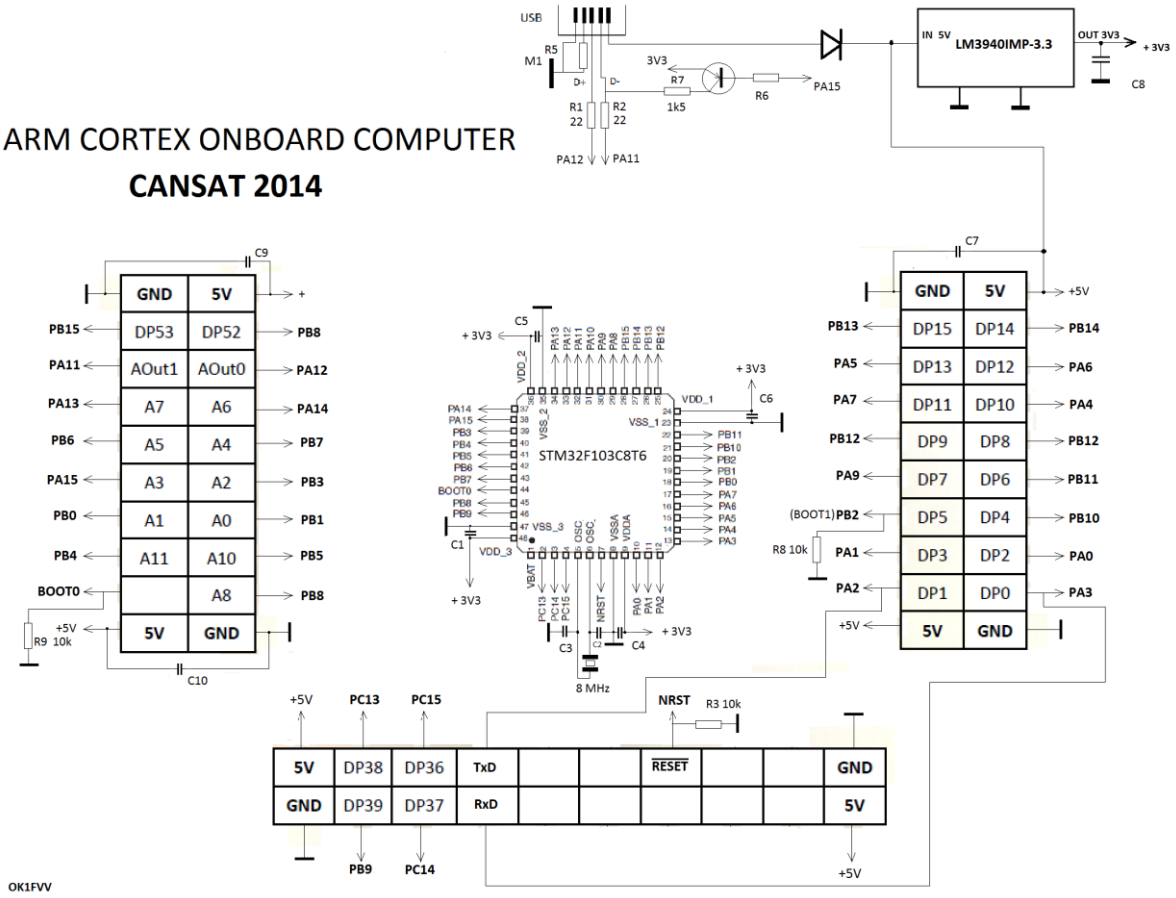


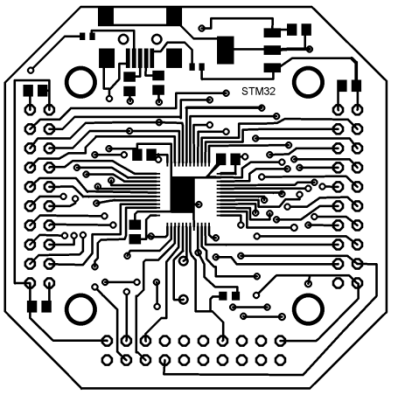


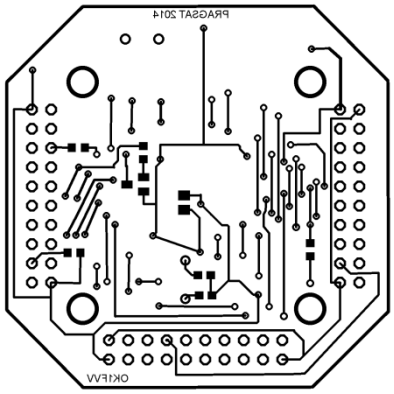


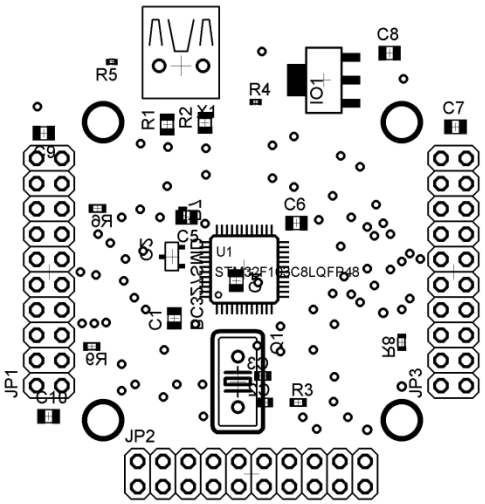


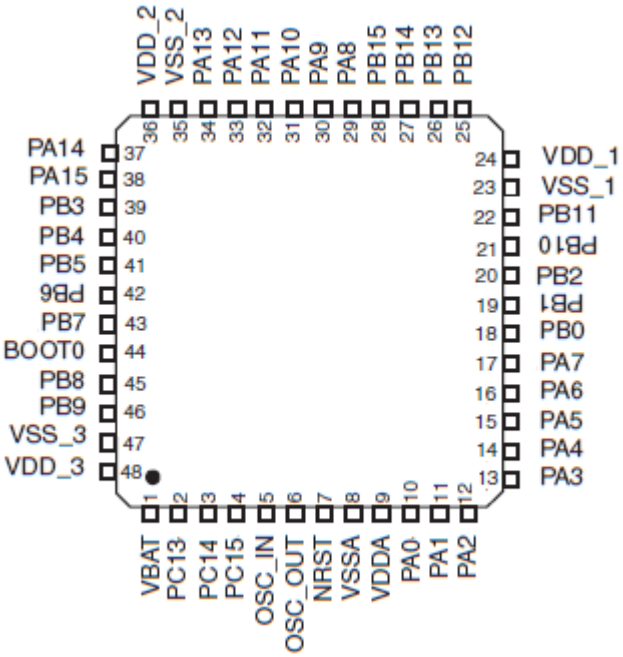
2.3 Vlastní počítač s ARM Cortex STM32



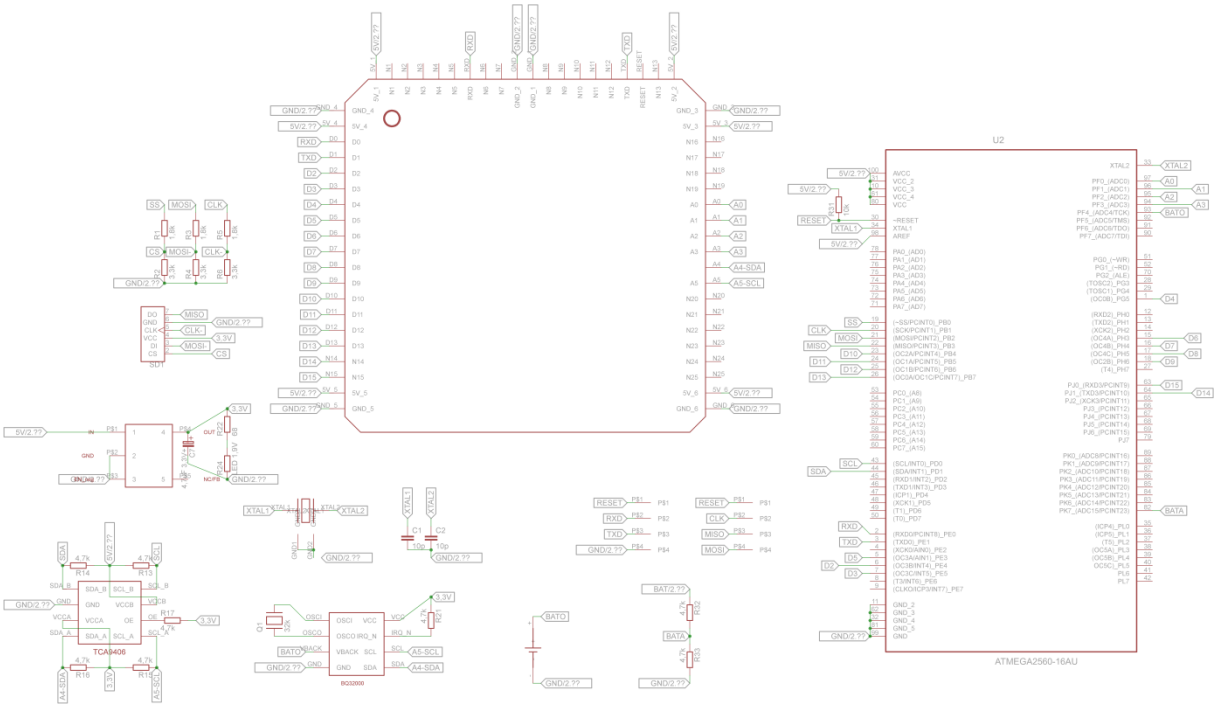


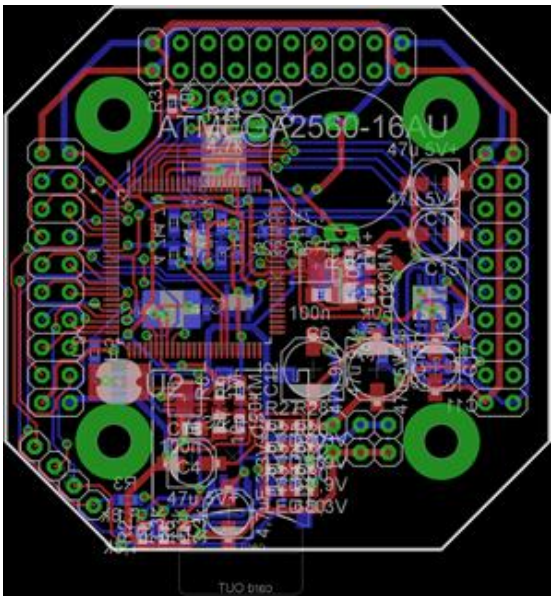


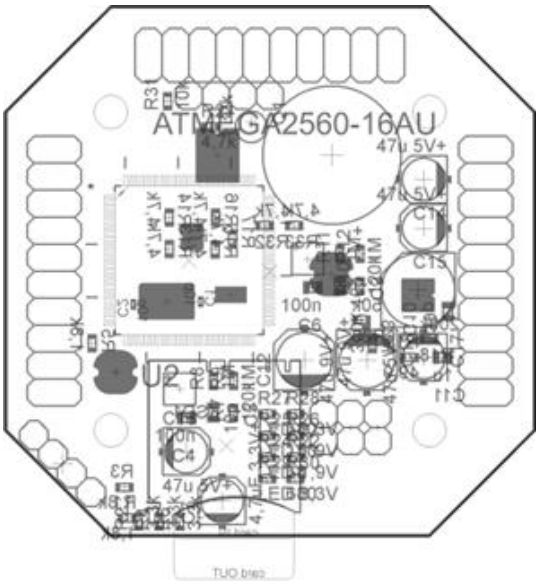


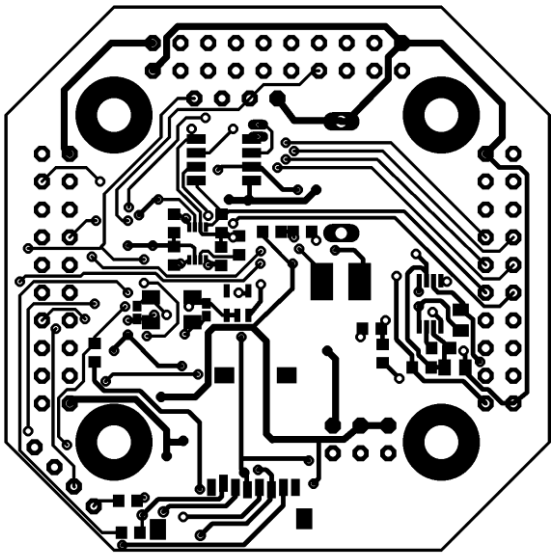


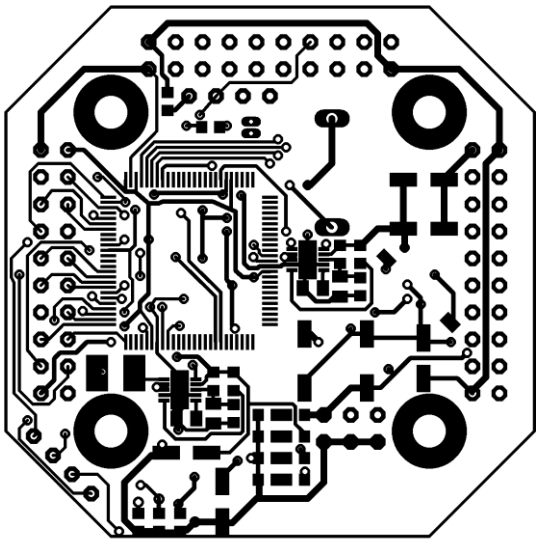
2.4 Palubní počítač navržený studenty











3. Čidla

Předpokládáme použití čidel tlaku, teploty, 3D akcelerometru, 3D magnetometru a 3D gyroskopu od firmy STMicroelectronics. Využijeme přitom modul STEVAL MKI124V1 module, který obsahuje LPS331AP (čidlo tlaku), LSM303DLHC (3D akcelerometr, 3D magnetometr) a L3GD20 (3D gyroscop) . S modulem se komunikuje prostřednictvím I2C. Protože neobsahuje teploměr, doplnil jsem tento modul destičkou s I2C čidlem teploty od STM, kompatibilním s LM75). Destička se přiletuje k MKI124V1 zespodu:

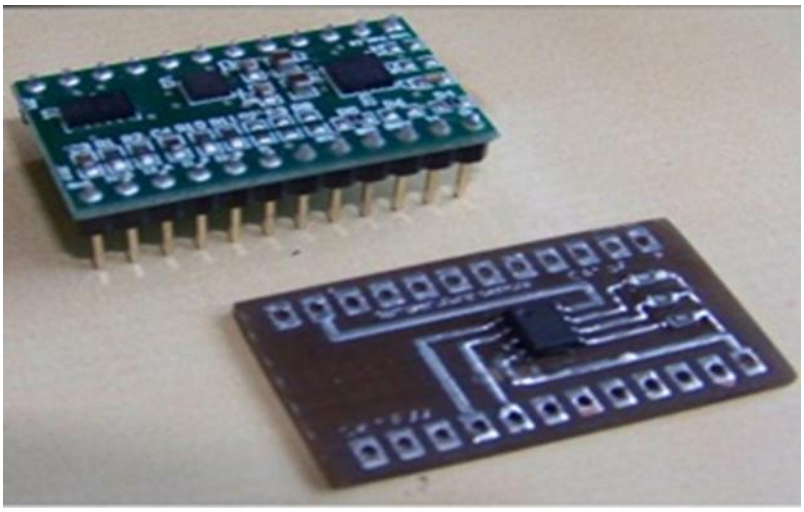
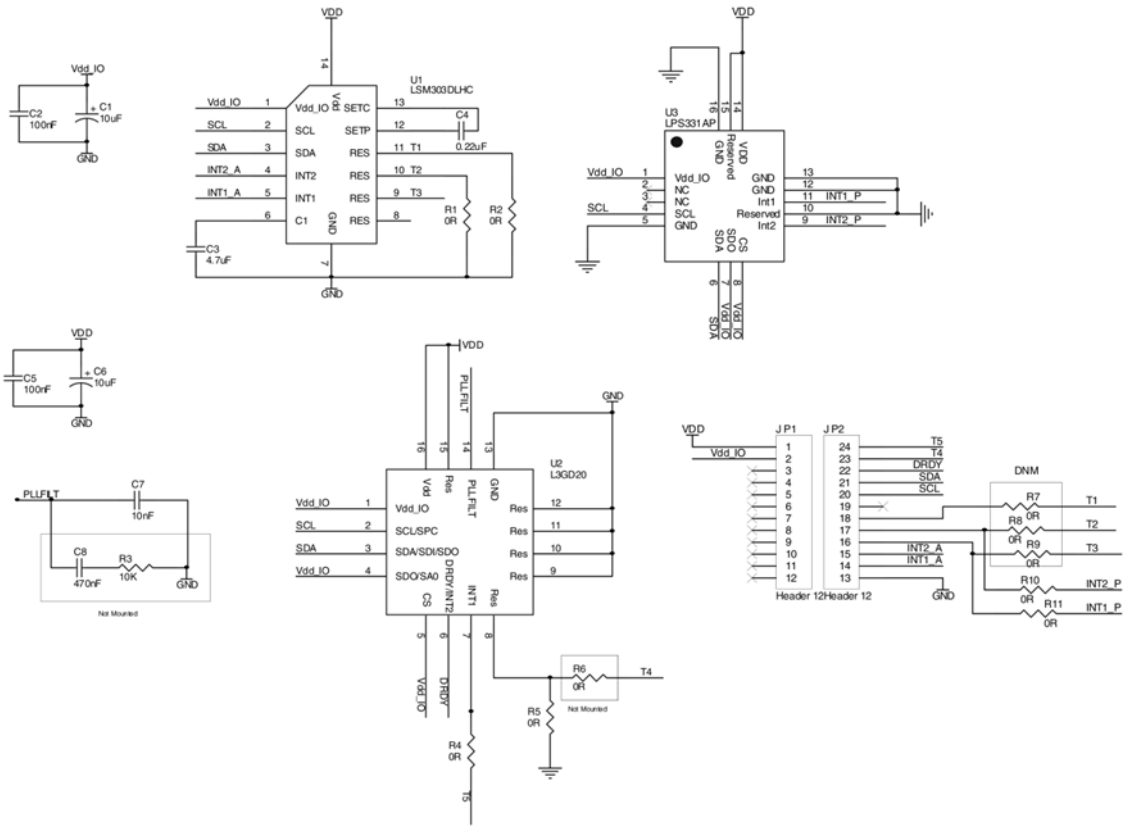
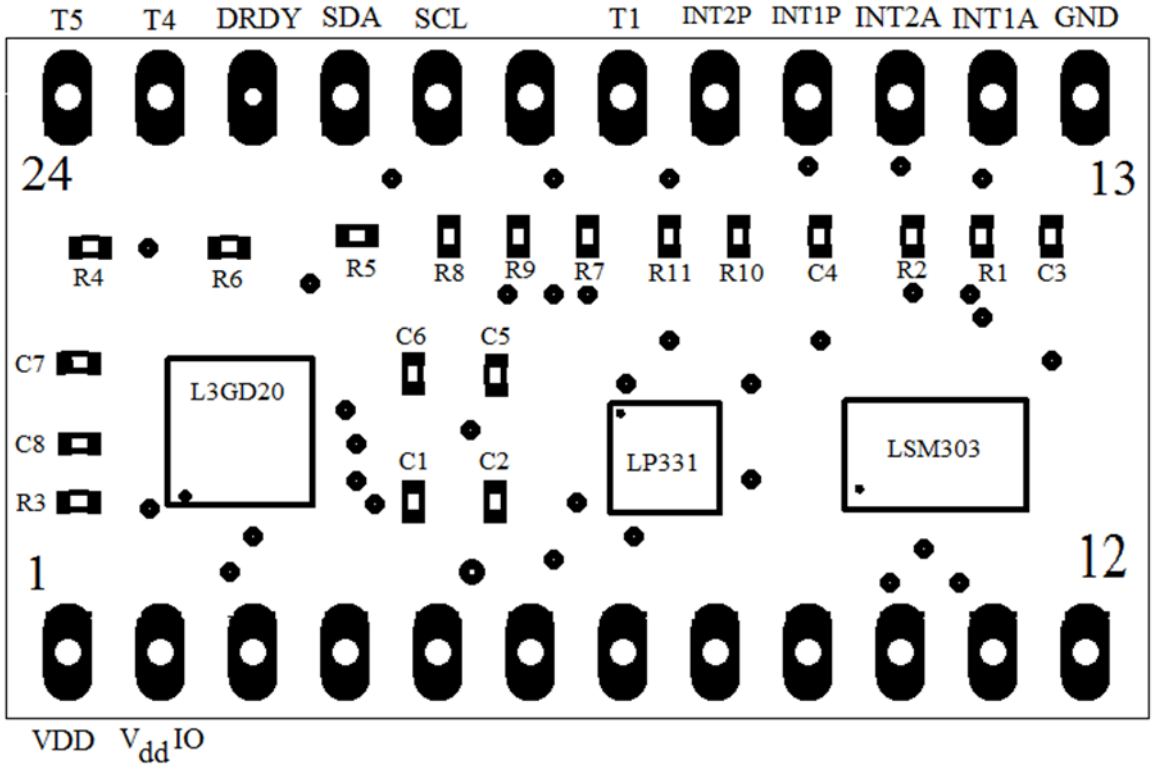


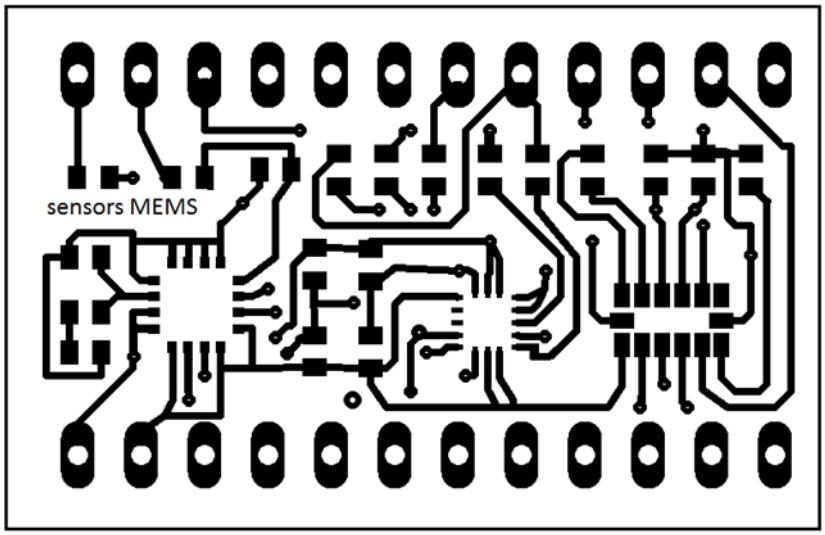
Schéma modulu MKI124V1:

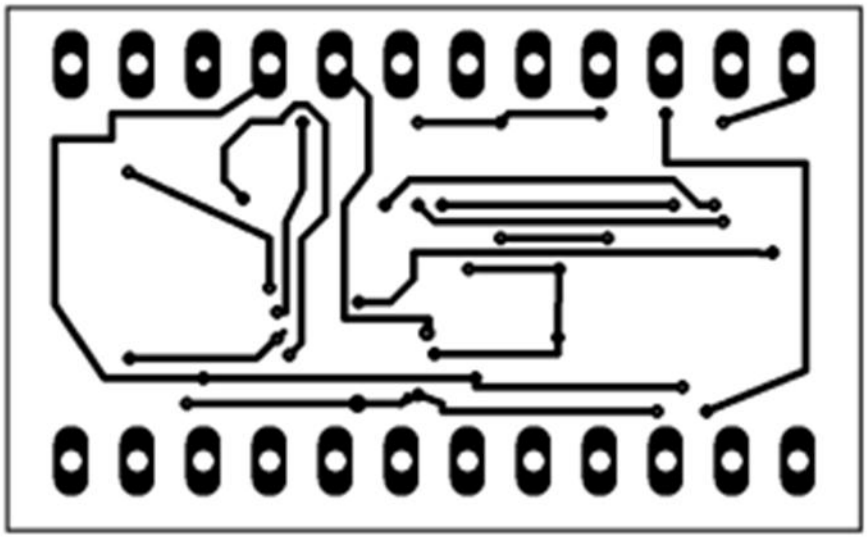


Rozložení součástí tohoto modulu:

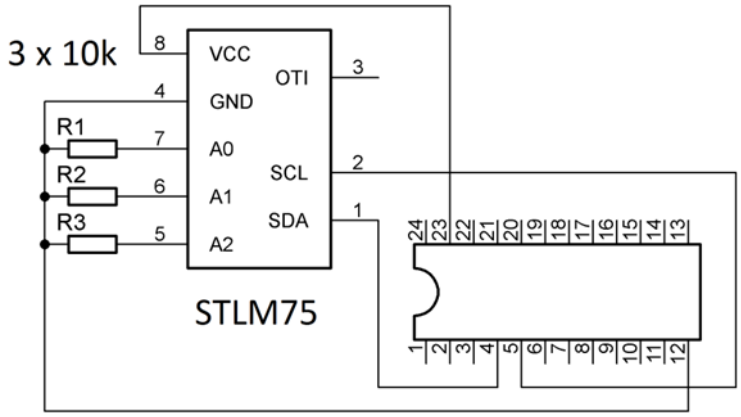

T5
T4
DRDY
SDA
SCL
T1
INT2P
INT1P
INT2A
INT1A
GND
24
13
R4
R6
R5
R8
R9
R7
R11
R10
C4
R2
R1
C3
C6
C5
C7
L3GD20
C8
C1
C2
LP331
LSM303
R3
1
12
VDD
V_{dd} IO

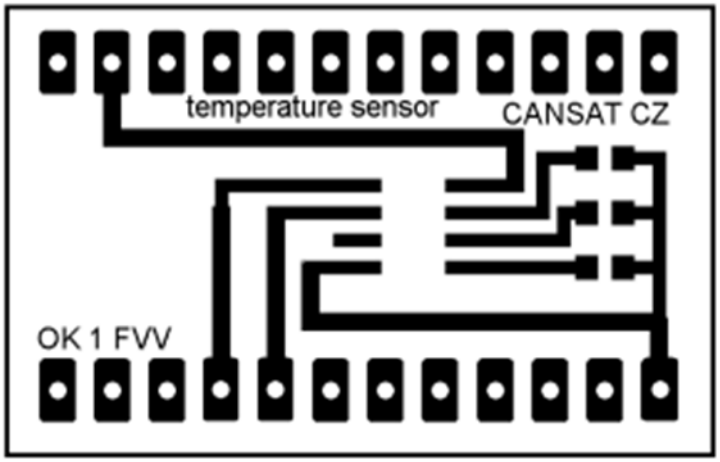
A ještě PCB:

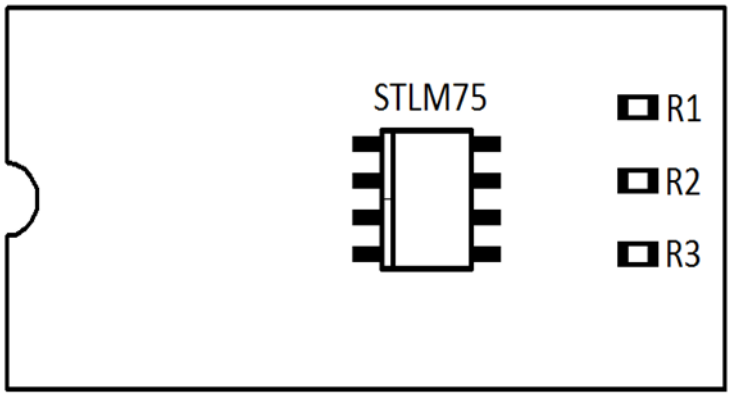

sensors MEMS



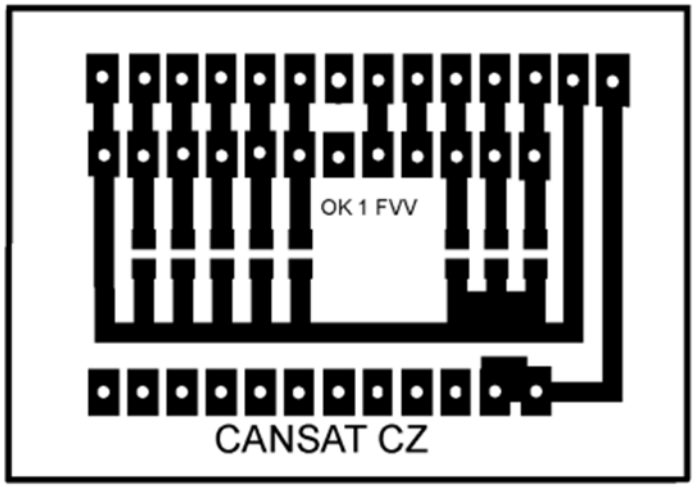
Zapojení destičky s čidlem teploty:

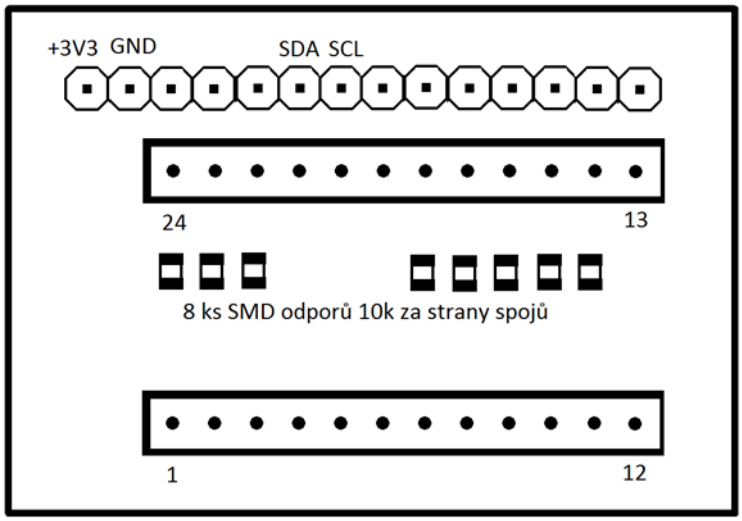






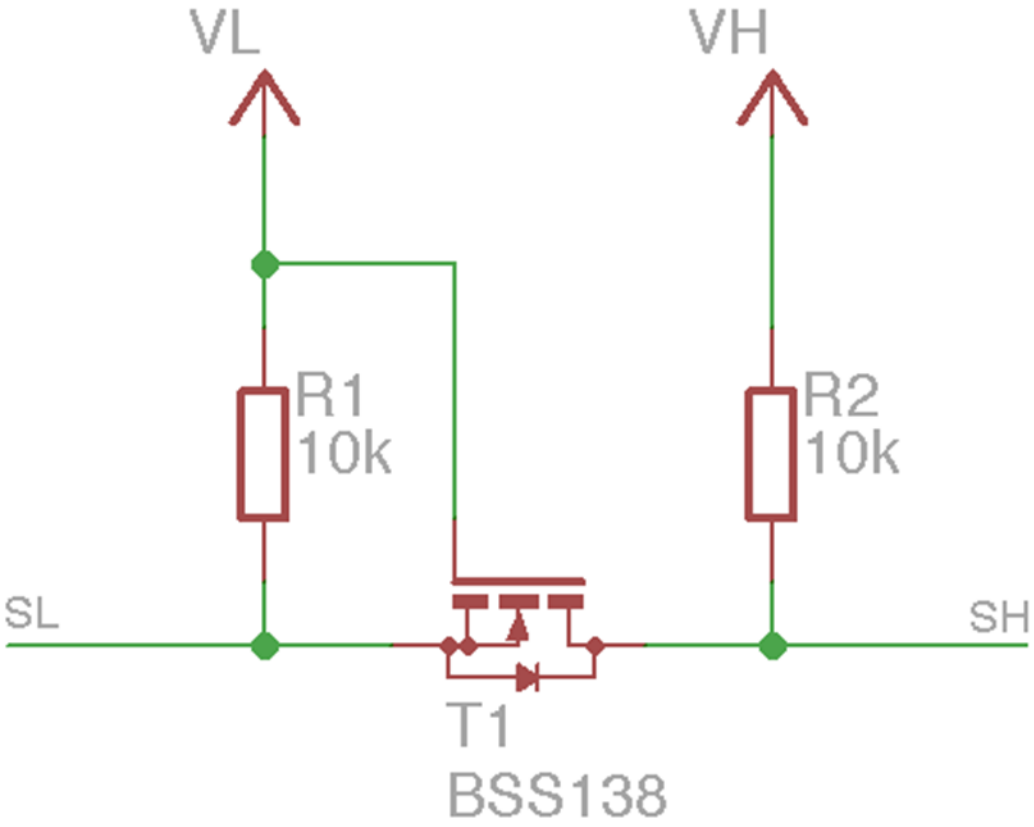
Na destičku s čidlem teploty by bylo dobré umístit ještě i2c obvod měřící čas (hodiny). Pro práci s adaptérovou destičkou STEVAL-MKI124V1 si ještě můžeme vyrobit pomocnou základovou destičku . Obsahuje jen sokl DIL24 pro adaptérovou destičku, 14 pinový konektor pro komunikaci s mikrořadičem a 8 smd odporů zajišťujících úroveň logické nuly na příslušných pinech čidel MEMS. Kromě dvou vodičů napájení 3,3V použijeme ke komunikaci





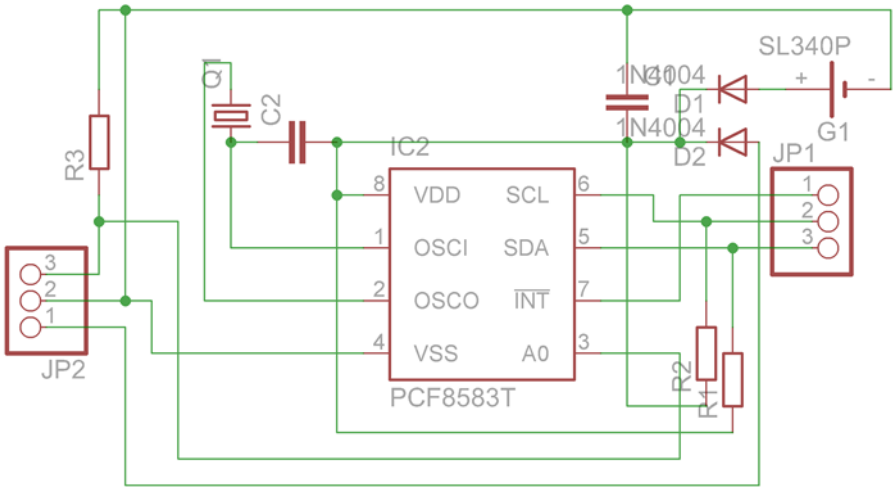
Ještě se podíváme na základní parametry MKI124V1 týkající se napájení:

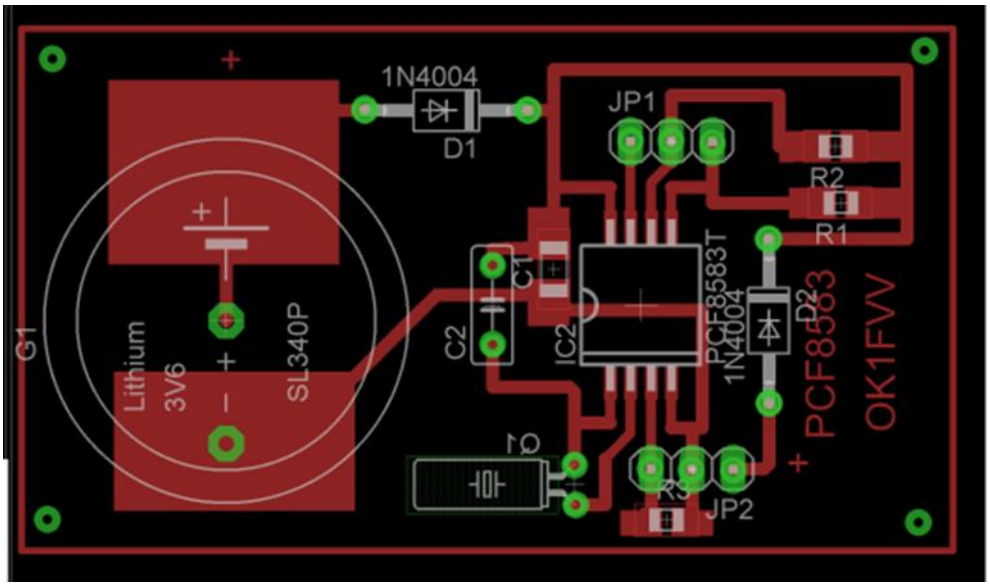
Napájení 2,4 až 3,6 V. I/O napěťové úrovně max. 3,6V takže **se nesmí připojit na TTL přímo,** tj ani k Arduino UNO, které je napájeno 5V !
Protože i2c signály SDA a SCL jsou obousměrné, musí být převodník mezi 3V3 a 5V také obousměrný. Je jím integrovaný obvod SN74LVC2T45. Lze však místo něj použít i zapojení s tranzistorem BSS138



Další poznámka:
Budeme měřit teplotu, tlak, 3D magnetické pole a 3D akceleraci, popř. další veličiny a to opakovaně (cca po 1 s). Nesmíme ale zapomenout, že budeme potřebovat také **informace o čase (time)**. Buď absolutní čas, nebo čas např. od reset programu a někde mít zaznamenán absolutní (skutečný) čas resetu (zahájení programu.

Měření času.
K měření času jsem odzkoušel I2C obvod PCF8583T (mají v GME). Jeho zapojení je:





Pro programovou obsluhu jsem odzkoušel následující sketch:

```
#include <Wire.h> // necessary, or the application won't build properly
#include <stdio.h>
#include <PCF8583.h>
/*************************************************************************
 *  read/write serial interface to PCF8583 RTC via I2C interface
 *
 *  Arduino analog input 5 - I2C SCL (PCF8583 pin 6)
 *  Arduino analog input 4 - I2C SDA (PCF8583 pin 5)
 *
 *  You can set the type by sending it YYMMddhhmmss;
 *  the semicolon on the end tells it you're done...
 *
 *************************************************************************/

int correct_address = 0;
PCF8583 p (0xA0);
void setup(void){
  Serial.begin(9600);
  Serial.print("booting...");
  Serial.println(" done");
     /* p.hour = 13;
      p.minute = 15;
      p.second = 0;
      p.year = 2014;
      p.month = 1;
      p.day = 25;
      p.set_time();
    */

}



void loop(void){
 if(Serial.available() > 0){
       p.year= (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48)) + 2000;
       p.month = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
```

```
      p.day = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
      p.hour  = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
      p.minute = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
      p.second = (byte) ((Serial.read() - 48) * 10 + (Serial.read() - 48)); // Use of
(byte) type casting and ascii math to achieve result.

      if(Serial.read() == ';'){
        Serial.println("setting date");
       p.set_time();
      }
  }


  p.get_time();
  char time[50];
  sprintf(time, "%02d/%02d/%02d %02d:%02d:%02d",
        p.year, p.month, p.day, p.hour, p.minute, p.second);
  Serial.println(time);

  delay(1000);
}
```

Potřebujeme k němu ještě knihovnu PCF8583:

PCF8583.h

```
/*

   Usage:
      PCF8583 p(0xA0);
      p.get_time();

      Serial.print("year: ");
      Serial.println(p.year);


      p.hour = 14;
      p.minute = 30
      p.second = 0
      p.year = 2014
      p.month = 1
      p.day = 19
      p.set_time();


*/

#ifndef PCF8583_H
#define PCF8583_H

#include <Arduino.h>
#include <../Wire/Wire.h>

class PCF8583 {
    int address;
    int dow;
  public:
    int second;
    int minute;
    int hour;
```

```
    int day;
    int month;
    int year;
    int year_base;

    int alarm_milisec;
    int alarm_second;
    int alarm_minute;
    int alarm_hour;
    int alarm_day;

    PCF8583(int device_address);
    void init ();

    void get_time();
    void set_time();
    void get_alarm();
    int  get_day_of_week() const {
      return dow;
    }

    void set_daily_alarm();
    int bcd_to_byte(byte bcd);
    byte int_to_bcd(int in);
};


#endif  //PCF8583_H
```

PCF8583.cpp

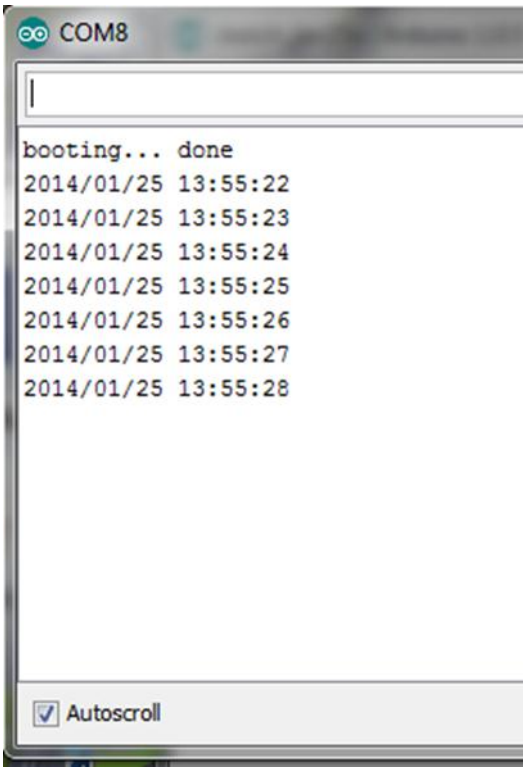
```
#include <Arduino.h>
#include <Wire.h>
#include "PCF8583.h"

namespace {
  bool IsLeapYear(int year) {
    return !(year % 400) || ((year % 100) && !(year % 4));
  }

  byte DayOfWeek(const PCF8583 &now) {
    static char PROGMEM MonthTable[24] = {0, 3, 3, 6, 1, 4, 6, 2, 5, 0, 3, 5, -1, 2,
3, 6, 1, 4, 6, 2, 5, 0, 3, 5};
    byte y = now.year % 100, c = 6 - 2 * ((now.year / 100) % 4);
    return (now.day + pgm_read_byte_near(MonthTable + IsLeapYear(now.year) * 12 +
now.month - 1) + y + (y / 4) + c) % 7;
  }
}

// provide device address as a full 8 bit address (like the datasheet)
PCF8583::PCF8583(int device_address) {
  address = device_address >> 1;  // convert to 7 bit so Wire doesn't choke
  Wire.begin();
}

// initialization
void PCF8583::init()
{

Wire.beginTransmission(address);
Wire.write(0x00);
```

```
Wire.write(0x04);   // Set alarm on int\ will turn to vcc
Wire.endTransmission();

}

void PCF8583::get_time(){

  Wire.beginTransmission(address);
  Wire.write(0xC0);   // stop counting, don't mask
  Wire.endTransmission();

  Wire.beginTransmission(address);
  Wire.write(0x02);
  Wire.endTransmission();
  Wire.requestFrom(address, 5);

  second = bcd_to_byte(Wire.read());
  minute = bcd_to_byte(Wire.read());
  hour   = bcd_to_byte(Wire.read());
  byte incoming = Wire.read(); // year/date counter
  day    = bcd_to_byte(incoming & 0x3f);
  year   = (int)((incoming >> 6) & 0x03);      // it will only hold 4 years...
  incoming = Wire.read();
  month  = bcd_to_byte(incoming & 0x1f);
  dow = incoming >> 5;

  //  but that's not all - we need to find out what the base year is
  //  so we can add the 2 bits we got above and find the real year
  Wire.beginTransmission(address);
  Wire.write(0x10);
  Wire.endTransmission();
  Wire.requestFrom(address, 2);
  year_base = 0;
  year_base = Wire.read();
  year_base = year_base << 8;
  year_base = year_base | Wire.read();
  year = year + year_base;
}


void PCF8583::set_time()
 {

  if (!IsLeapYear(year) && 2 == month && 29 == day) {
    month = 3;
    day = 1;
  }

  // Attempt to find the previous leap year
  year_base = year - year % 4;
  if (!IsLeapYear(year_base)) {
    // Not a leap year (new century), make sure the calendar won't use a 29 days
February.
    year_base = year - 1;
  }

  dow = DayOfWeek(*this);

  Wire.beginTransmission(address);
  Wire.write(0xC0);   // stop counting, don't mask
  Wire.endTransmission();
```

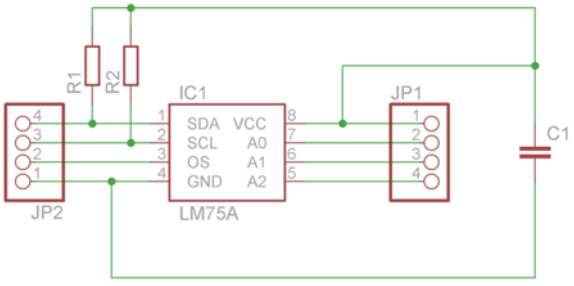
```
  Wire.beginTransmission(address);
  Wire.write(0x02);
  Wire.write(int_to_bcd(second));
  Wire.write(int_to_bcd(minute));
  Wire.write(int_to_bcd(hour));
  Wire.write(((byte)(year - year_base) << 6) | int_to_bcd(day));
  Wire.write((dow << 5) | (int_to_bcd(month) & 0x1f));
  Wire.endTransmission();

  Wire.beginTransmission(address);
  Wire.write(0x10);
  Wire.write(year_base >> 8);
  Wire.write(year_base & 0x00ff);
  Wire.endTransmission();

  init(); // re set the control/status register to 0x04

  }

//Get the alarm at 0x09 adress
void PCF8583::get_alarm()
{
Wire.beginTransmission(address);
Wire.write(0x0A); // Set the register pointer to (0x0A)
Wire.endTransmission();

Wire.requestFrom(address, 4); // Read 4 values

alarm_second = bcd_to_byte(Wire.read());
alarm_minute = bcd_to_byte(Wire.read());
alarm_hour   = bcd_to_byte(Wire.read());

Wire.beginTransmission(address);
Wire.write(0x0E);
Wire.endTransmission();

Wire.requestFrom(address, 1); // Read weekday value

alarm_day = bcd_to_byte(Wire.read());
}

//Set a daily alarm
void PCF8583::set_daily_alarm()
{
Wire.beginTransmission(address);
Wire.write(0x08);
Wire.write(0x90);  // daily alarm set
Wire.endTransmission();

Wire.beginTransmission(address);
Wire.write(0x09); // Set the register pointer to (0x09)
Wire.write(0x00); // Set 00 at milisec
Wire.write(int_to_bcd(alarm_second));
Wire.write(int_to_bcd(alarm_minute));
Wire.write(int_to_bcd(alarm_hour));
Wire.write(0x00); // Set 00 at day
Wire.endTransmission();
}

int PCF8583::bcd_to_byte(byte bcd){
  return ((bcd >> 4) * 10) + (bcd & 0x0f);
}
```

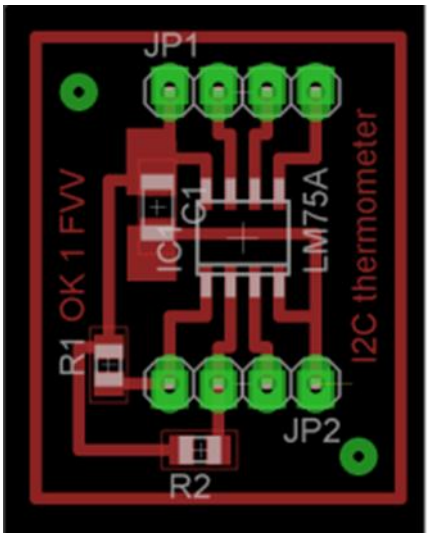
```
byte PCF8583::int_to_bcd(int in){
  return ((in / 10) << 4) + (in % 10);
}
```

Výstup tohoto programu přes sériový port na monitoru je např.



Měření teploty.
Odzkoušel jsem LM75, LM76 a LM92. Pro LM75 vezmeme např. zapojení:





Pro programovou obsluhu jsem odzkoušel následující sketch:

```
#include <Wire.h>
#include <LM75.h>

LM75 sensor;  // initialize an LM75 object
// You can also initiate with another address as follows:
//LM75 sensor(LM75_ADDRESS | 0b001);  // if A0->GND, A1->GND and A2->Vcc

void setup()
{
  Wire.begin();
  Serial.begin(9600);
}

void loop()
{
  // get temperature from sensor
  Serial.print("Current temp: ");
  Serial.print(sensor.temp());
  Serial.println(" C");

  // Tos Set-point
  //sensor.tos(47.5); // set at 47.5'C
  //Serial.print("Tos set at ");
  //Serial.print(sensor.tos());
  //Serial.println(" C");

  // Thyst Set-point
  //sensor.thyst(42); // set at 42'C
  //Serial.print("Thyst set at ");
  //Serial.print(sensor.thyst());
  //Serial.println(" C");

  // shutdown the sensor and wait a while
  sensor.shutdown(true);
  delay(1000);
  // wake up sensor for next time around
  sensor.shutdown(false);

  Serial.println();
```

```
}
```

Potřebujeme k němu ještě knihovnu LM75:

LM75.h

```
#ifndef LM75_h

#define LM75_h

#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

#define LM75_ADDRESS 0x48

#define LM75_TEMP_REGISTER 0
#define LM75_CONF_REGISTER 1
#define LM75_THYST_REGISTER 2
#define LM75_TOS_REGISTER 3

#define LM75_CONF_SHUTDOWN  0
#define LM75_CONF_OS_COMP_INT 1
#define LM75_CONF_OS_POL 2
#define LM75_CONF_OS_F_QUE 3

class LM75 {
    int address;
    word float2regdata (float);
    float regdata2float (word);
    word _register16 (byte);
    void _register16 (byte, word);
    word _register8 (byte);
    void _register8 (byte, byte);
  public:
    LM75 ();
    LM75 (byte);
    float temp (void);
    byte conf (void);
    void conf (byte);
    float tos (void);
    void tos (float);
    float thyst (void);
    void thyst (float);
    void shutdown (boolean);
    boolean shutdown (void);
};

#endif
```

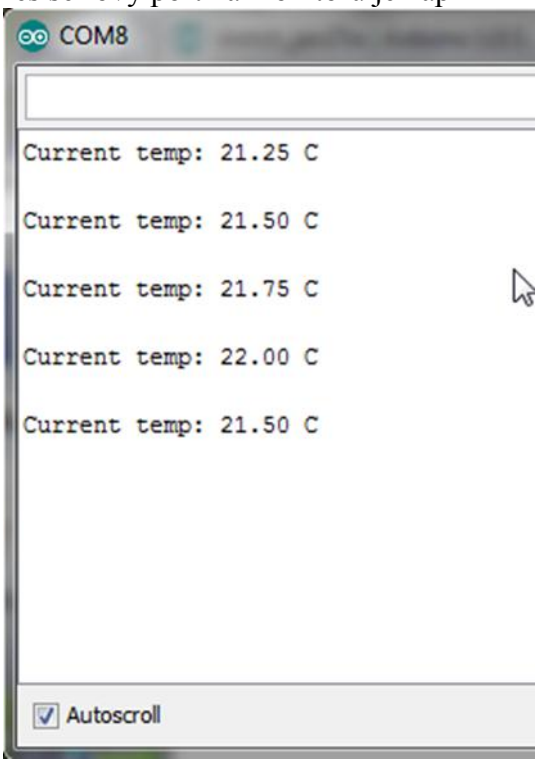
LM75.cpp

```
#include <Wire.h>
#include "LM75.h"

LM75::LM75 () {
  address = LM75_ADDRESS;
```

```
}

LM75::LM75 (byte addr) {
  address = addr;
}

word LM75::float2regdata (float temp)
{
  // First multiply by 8 and coerce to integer to get +/- whole numbers
  // Then coerce to word and bitshift 5 to fill out MSB
  return (word)((int)(temp * 8) << 5);
}
float LM75::regdata2float (word regdata)
{
  return ((float)(int)regdata / 32) / 8;
}word LM75::_register16 (byte reg) {
  Wire.beginTransmission(address);
  Wire.write(reg);
  Wire.endTransmission();

  Wire.requestFrom(address, 2);
  word regdata = (Wire.read() << 8) | Wire.read();
  return regdata;
}

void LM75::_register16 (byte reg, word regdata) {
  byte msb = (byte)(regdata >> 8);
  byte lsb = (byte)(regdata);

  Wire.beginTransmission(address);
  Wire.write(reg);
  Wire.write(msb);
  Wire.write(lsb);
  Wire.endTransmission();
}

word LM75::_register8 (byte reg) {
  Wire.beginTransmission(address);
  Wire.write(reg);
  Wire.endTransmission();

  Wire.requestFrom(address, 1);
  return Wire.read();
}

void LM75::_register8 (byte reg, byte regdata) {
  Wire.beginTransmission(address);
  Wire.write(reg);
  Wire.write(regdata);
  Wire.endTransmission();
}

float LM75::temp (void) {
  return regdata2float(_register16(LM75_TEMP_REGISTER));
}

byte LM75::conf () {
  return _register8(LM75_CONF_REGISTER);
}

void LM75::conf (byte data) {
  _register8(LM75_CONF_REGISTER, data);
```

```
}

float LM75::tos () {
  return regdata2float(_register16(LM75_TOS_REGISTER));
}

void LM75::tos (float temp) {
  _register16(LM75_TOS_REGISTER, float2regdata(temp));
}

float LM75::thyst () {
  return regdata2float(_register16(LM75_THYST_REGISTER));
}

void LM75::thyst (float temp) {
  _register16(LM75_THYST_REGISTER, float2regdata(temp));
}

boolean LM75::shutdown () {
  return conf() & 0x01;
}

void LM75::shutdown (boolean val) {
  conf(val << LM75_CONF_SHUTDOWN);
}
```

Výstup tohoto programu přes sériový port na monitoru je např.



Pozn.:
U LM75 je

UPPER BYTE								LOWER BYTE							
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Sign bit 1= Negative 0 = Positive	MSB 64˚C	32°C	16°C	8°C	4°C	2°C	1°C	LSB 0.5°C	X	X	X	X	X	X	X

X = Don't care.

U LM76 a LM92 je.:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Sign	MSB	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	CRIT	HIGH	LOW
													Status Bits		

D0–D2: Status Bits

D3–D15: Temperature Data. One LSB = 0.0625°C. Two's complement format.

Proto upravíme LM75.cpp

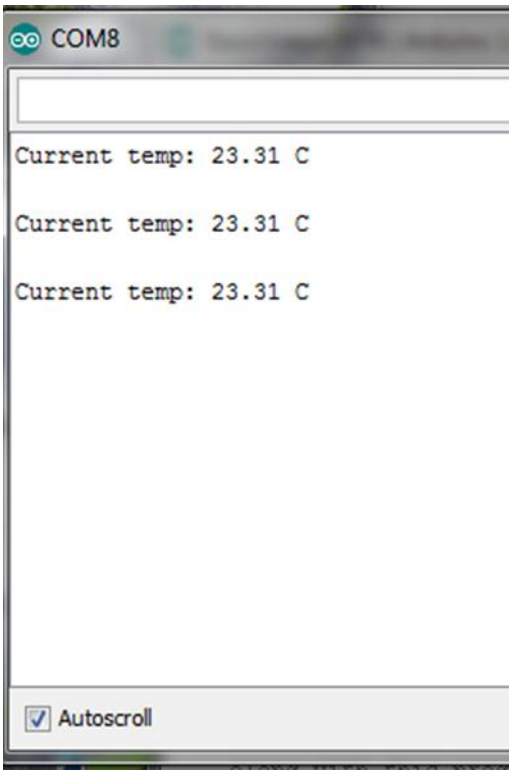
```
float LM75::regdata2float (word regdata)
{
  return ((float)(int)regdata / 32) / 8;
}
```

Na LM76.cpp, která je stejná až na

```
float LM76::regdata2float (word regdata)
{
  return ((float)(int)regdata / 32) / 4;
}
```

Pochopitelně jsme kromě toho všude přepsali LM75 na LM76 a soubor označili LM76.cpp

Obdobně LM76.h.
Výsledkem bude:



Měření zrychlení.

Použil jsem MEMS z STEVAL-MKI124V1. Protože Arduino UNO je na TTL (5V logice) a STEVAL-MKI124V1 na 3V3, použil jsem pro dva vodiče i2c převod s polem řízenými tranzistory.
Pro programovou obsluhu jsem odzkoušel následující sketch:

```
#include <Wire.h>
#include <Sensor.h>
#include <LSM303_U.h>

/* Assign a unique ID to this sensor at the same time */
LSM303_Accel_Unified accel = LSM303_Accel_Unified(54321);

void displaySensorDetails(void)
{
  sensor_t sensor;
  accel.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println("
m/s^2");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println("
m/s^2");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println("
m/s^2");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

void setup(void)
```

```
{
  Serial.begin(9600);
  Serial.println("Accelerometer Test"); Serial.println("");

  /* Initialise the sensor */
  if(!accel.begin())
  {
    /* There was a problem detecting the ADXL345 ... check your connections */
    Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
    while(1);
  }

  /* Display some basic information on this sensor */
  displaySensorDetails();
}

void loop(void)
{
  /* Get a new sensor event */
  sensors_event_t event;
  accel.getEvent(&event);

  /* Display the results (acceleration is measured in m/s^2) */
  Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event.acceleration.y); Serial.print("  ");
  Serial.print("Z: "); Serial.print(event.acceleration.z); Serial.print("
");Serial.println("m/s^2 ");
  delay(500);
}
```

Potřebujeme k němu ještě knihovny Sensor a LSM303_U:

Sensor.h

```
#ifndef SENSOR_H
#define SENSOR_H

#if ARDUINO >= 100
 #include "Arduino.h"
 #include "Print.h"
#else
 #include "WProgram.h"
#endif

/* Intentionally modeled after sensors.h in the Android API:
 *
https://github.com/android/platform_hardware_libhardware/blob/master/include/hardware/
sensors.h */

/* Constants */
#define SENSORS_GRAVITY_EARTH             (9.80665F)              /**< Earth's gravity
in m/s^2 */
#define SENSORS_GRAVITY_MOON              (1.6F)                  /**< The moon's
gravity in m/s^2 */
#define SENSORS_GRAVITY_SUN               (275.0F)                /**< The sun's
gravity in m/s^2 */
#define SENSORS_GRAVITY_STANDARD          (SENSORS_GRAVITY_EARTH)
#define SENSORS_MAGFIELD_EARTH_MAX        (60.0F)                 /**< Maximum
magnetic field on Earth's surface */
#define SENSORS_MAGFIELD_EARTH_MIN        (30.0F)                 /**< Minimum
magnetic field on Earth's surface */
```

```
#define SENSORS_PRESSURE_SEALEVELHPA      (1013.25F)             /**< Average sea
level pressure is 1013.25 hPa */
#define SENSORS_DPS_TO_RADS               (0.017453293F)         /**< Degrees/s to
rad/s multiplier */
#define SENSORS_GAUSS_TO_MICROTESLA       (100)                  /**< Gauss to micro-
Tesla multiplier */

/** Sensor types */
typedef enum
{
  SENSOR_TYPE_ACCELEROMETER         = (1),   /**< Gravity + linear acceleration */
  SENSOR_TYPE_MAGNETIC_FIELD        = (2),
  SENSOR_TYPE_ORIENTATION           = (3),
  SENSOR_TYPE_GYROSCOPE             = (4),
  SENSOR_TYPE_LIGHT                 = (5),
  SENSOR_TYPE_PRESSURE              = (6),
  SENSOR_TYPE_PROXIMITY             = (8),
  SENSOR_TYPE_GRAVITY               = (9),
  SENSOR_TYPE_LINEAR_ACCELERATION   = (10),  /**< Acceleration not including gravity
*/
  SENSOR_TYPE_ROTATION_VECTOR       = (11),
  SENSOR_TYPE_RELATIVE_HUMIDITY     = (12),
  SENSOR_TYPE_AMBIENT_TEMPERATURE   = (13),
  SENSOR_TYPE_VOLTAGE               = (15),
  SENSOR_TYPE_CURRENT               = (16),
  SENSOR_TYPE_COLOR                 = (17)
} sensors_type_t;

/** struct sensors_vec_s is used to return a vector in a common format. */
typedef struct {
    union {
        float v[3];
        struct {
            float x;
            float y;
            float z;
        };
        /* Orientation sensors */
        struct {
            float roll;    /**< Rotation around the longitudinal axis (the plane body,
'X axis'). Roll is positive and increasing when moving downward. -90°<=roll<=90° */
            float pitch;   /**< Rotation around the lateral axis (the wing span, 'Y
axis'). Pitch is positive and increasing when moving upwards. -180°<=pitch<=180°) */
            float heading; /**< Angle between the longitudinal axis (the plane body)
and magnetic north, measured clockwise when viewing from the top of the device. 0-359°
*/
        };
    };
    int8_t status;
    uint8_t reserved[3];
} sensors_vec_t;

/** struct sensors_color_s is used to return color data in a common format. */
typedef struct {
    union {
        float c[3];
        /* RGB color space */
        struct {
            float r;       /**< Red component */
            float g;       /**< Green component */
            float b;       /**< Blue component */
        };
```

```
    };
    uint32_t rgba;         /**< 24-bit RGBA value */
} sensors_color_t;

/* Sensor event (36 bytes) */
/** struct sensor_event_s is used to provide a single sensor event in a common format.
*/
typedef struct
{
    int32_t version;                        /**< must be sizeof(struct
sensors_event_t) */
    int32_t sensor_id;                      /**< unique sensor identifier */
    int32_t type;                           /**< sensor type */
    int32_t reserved0;                      /**< reserved */
    int32_t timestamp;                      /**< time is in milliseconds */
    union
    {
        float           data[4];
        sensors_vec_t   acceleration;       /**< acceleration values are in meter
per second per second (m/s^2) */
        sensors_vec_t   magnetic;           /**< magnetic vector values are in
micro-Tesla (uT) */
        sensors_vec_t   orientation;        /**< orientation values are in degrees
*/
        sensors_vec_t   gyro;               /**< gyroscope values are in rad/s */
        float           temperature;        /**< temperature is in degrees
centigrade (Celsius) */
        float           distance;           /**< distance in centimeters */
        float           light;              /**< light in SI lux units */
        float           pressure;           /**< pressure in hectopascal (hPa) */
        float           relative_humidity;  /**< relative humidity in percent */
        float           current;            /**< current in milliamps (mA) */
        float           voltage;            /**< voltage in volts (V) */
        sensors_color_t color;              /**< color in RGB component values */
    };
} sensors_event_t;

/* Sensor details (40 bytes) */
/** struct sensor_s is used to describe basic information about a specific sensor. */
typedef struct
{
    char     name[12];                      /**< sensor name */
    int32_t  version;                       /**< version of the hardware + driver */
    int32_t  sensor_id;                     /**< unique sensor identifier */
    int32_t  type;                          /**< this sensor's type (ex.
SENSOR_TYPE_LIGHT) */
    float    max_value;                     /**< maximum value of this sensor's
value in SI units */
    float    min_value;                     /**< minimum value of this sensor's
value in SI units */
    float    resolution;                    /**< smallest difference between two
values reported by this sensor */
    int32_t  min_delay;                     /**< min delay in microseconds between
events. zero = not a constant rate */
} sensor_t;

class Sensor {
 public:
  // Constructor(s)
  // Sensor();
  void constructor();
```

```
  // These must be defined by the subclass
  virtual void getEvent(sensors_event_t*);
  virtual void getSensor(sensor_t*);
};

#endif
```

Sensor.cpp

```
#include "Sensor.h"
#include <avr/pgmspace.h>

void Sensor::constructor() {
}
```

LSM303_U.h

```
#ifndef __LSM303_H__
#define __LSM303_H__

#if (ARDUINO >= 100)
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif

#include <Sensor.h>
#include <Wire.h>

/*=========================================================================
    I2C ADDRESS/BITS
    -----------------------------------------------------------------------*/
    #define LSM303_ADDRESS_ACCEL          (0x32 >> 1)         // 0011001x
    #define LSM303_ADDRESS_MAG            (0x3C >> 1)         // 0011110x
/*=========================================================================*/

/*=========================================================================
    REGISTERS
    -----------------------------------------------------------------------*/
    typedef enum
    {                                                   // DEFAULT    TYPE
      LSM303_REGISTER_ACCEL_CTRL_REG1_A         = 0x20,   // 00000111   rw
      LSM303_REGISTER_ACCEL_CTRL_REG2_A         = 0x21,   // 00000000   rw
      LSM303_REGISTER_ACCEL_CTRL_REG3_A         = 0x22,   // 00000000   rw
      LSM303_REGISTER_ACCEL_CTRL_REG4_A         = 0x23,   // 00000000   rw
      LSM303_REGISTER_ACCEL_CTRL_REG5_A         = 0x24,   // 00000000   rw
      LSM303_REGISTER_ACCEL_CTRL_REG6_A         = 0x25,   // 00000000   rw
      LSM303_REGISTER_ACCEL_REFERENCE_A         = 0x26,   // 00000000   r
      LSM303_REGISTER_ACCEL_STATUS_REG_A        = 0x27,   // 00000000   r
      LSM303_REGISTER_ACCEL_OUT_X_L_A           = 0x28,
      LSM303_REGISTER_ACCEL_OUT_X_H_A           = 0x29,
      LSM303_REGISTER_ACCEL_OUT_Y_L_A           = 0x2A,
      LSM303_REGISTER_ACCEL_OUT_Y_H_A           = 0x2B,
      LSM303_REGISTER_ACCEL_OUT_Z_L_A           = 0x2C,
      LSM303_REGISTER_ACCEL_OUT_Z_H_A           = 0x2D,
      LSM303_REGISTER_ACCEL_FIFO_CTRL_REG_A     = 0x2E,
      LSM303_REGISTER_ACCEL_FIFO_SRC_REG_A      = 0x2F,
      LSM303_REGISTER_ACCEL_INT1_CFG_A          = 0x30,
      LSM303_REGISTER_ACCEL_INT1_SOURCE_A       = 0x31,
```

```
      LSM303_REGISTER_ACCEL_INT1_THS_A          = 0x32,
      LSM303_REGISTER_ACCEL_INT1_DURATION_A     = 0x33,
      LSM303_REGISTER_ACCEL_INT2_CFG_A          = 0x34,
      LSM303_REGISTER_ACCEL_INT2_SOURCE_A       = 0x35,
      LSM303_REGISTER_ACCEL_INT2_THS_A          = 0x36,
      LSM303_REGISTER_ACCEL_INT2_DURATION_A     = 0x37,
      LSM303_REGISTER_ACCEL_CLICK_CFG_A         = 0x38,
      LSM303_REGISTER_ACCEL_CLICK_SRC_A         = 0x39,
      LSM303_REGISTER_ACCEL_CLICK_THS_A         = 0x3A,
      LSM303_REGISTER_ACCEL_TIME_LIMIT_A        = 0x3B,
      LSM303_REGISTER_ACCEL_TIME_LATENCY_A      = 0x3C,
      LSM303_REGISTER_ACCEL_TIME_WINDOW_A       = 0x3D
    } lsm303AccelRegisters_t;

    typedef enum
    {
      LSM303_REGISTER_MAG_CRA_REG_M             = 0x00,
      LSM303_REGISTER_MAG_CRB_REG_M             = 0x01,
      LSM303_REGISTER_MAG_MR_REG_M              = 0x02,
      LSM303_REGISTER_MAG_OUT_X_H_M             = 0x03,
      LSM303_REGISTER_MAG_OUT_X_L_M             = 0x04,
      LSM303_REGISTER_MAG_OUT_Z_H_M             = 0x05,
      LSM303_REGISTER_MAG_OUT_Z_L_M             = 0x06,
      LSM303_REGISTER_MAG_OUT_Y_H_M             = 0x07,
      LSM303_REGISTER_MAG_OUT_Y_L_M             = 0x08,
      LSM303_REGISTER_MAG_SR_REG_Mg             = 0x09,
      LSM303_REGISTER_MAG_IRA_REG_M             = 0x0A,
      LSM303_REGISTER_MAG_IRB_REG_M             = 0x0B,
      LSM303_REGISTER_MAG_IRC_REG_M             = 0x0C,
      LSM303_REGISTER_MAG_TEMP_OUT_H_M          = 0x31,
      LSM303_REGISTER_MAG_TEMP_OUT_L_M          = 0x32
    } lsm303MagRegisters_t;
/*=========================================================================*/

/*=========================================================================
    MAGNETOMETER GAIN SETTINGS
    -----------------------------------------------------------------------*/
    typedef enum
    {
      LSM303_MAGGAIN_1_3                        = 0x20,  // +/- 1.3
      LSM303_MAGGAIN_1_9                        = 0x40,  // +/- 1.9
      LSM303_MAGGAIN_2_5                        = 0x60,  // +/- 2.5
      LSM303_MAGGAIN_4_0                        = 0x80,  // +/- 4.0
      LSM303_MAGGAIN_4_7                        = 0xA0,  // +/- 4.7
      LSM303_MAGGAIN_5_6                        = 0xC0,  // +/- 5.6
      LSM303_MAGGAIN_8_1                        = 0xE0   // +/- 8.1
    } lsm303MagGain;
/*=========================================================================*/

/*=========================================================================
    INTERNAL MAGNETOMETER DATA TYPE
    -----------------------------------------------------------------------*/
    typedef struct lsm303MagData_s
    {
        float x;
        float y;
        float z;
      float orientation;
    } lsm303MagData;
/*=========================================================================*/

/*=========================================================================
```

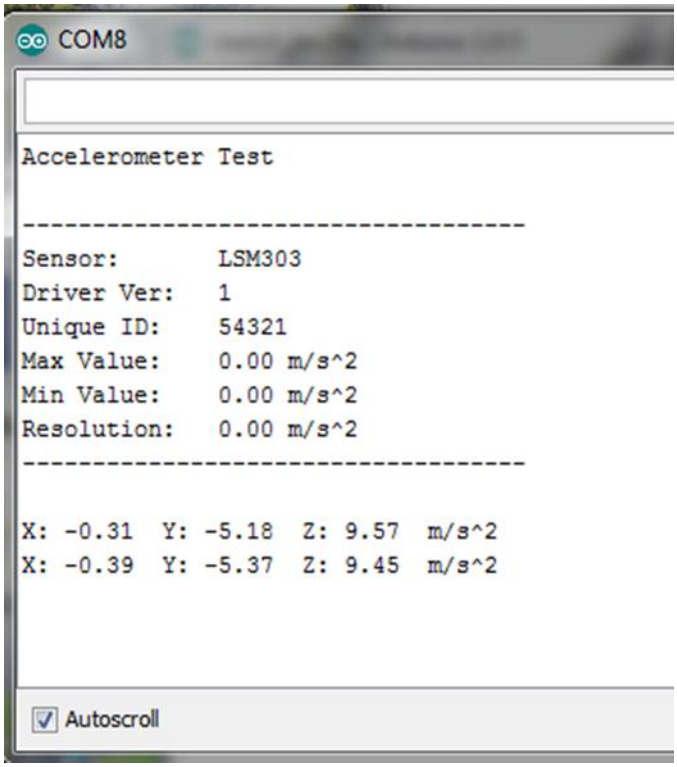
```
    INTERNAL ACCELERATION DATA TYPE
    ---------------------------------------------------------------------*/
    typedef struct lsm303AccelData_s
    {
      float x;
      float y;
      float z;
    } lsm303AccelData;
/*=========================================================================*/

/*=========================================================================
    CHIP ID
    ---------------------------------------------------------------------*/
    #define LSM303_ID                     (0b11010100)
/*=========================================================================*/

/* Unified sensor driver for the accelerometer */
class LSM303_Accel_Unified : public Sensor
{
  public:
    LSM303_Accel_Unified(int32_t sensorID = -1);

    bool begin(void);
    void getEvent(sensors_event_t*);
    void getSensor(sensor_t*);

  private:
    lsm303AccelData _accelData;   // Last read accelerometer data will be available
here
    int32_t         _sensorID;

    void write8(byte address, byte reg, byte value);
    byte read8(byte address, byte reg);
    void read(void);
};

/* Unified sensor driver for the magnetometer */
class LSM303_Mag_Unified : public Sensor
{
  public:
    LSM303_Mag_Unified(int32_t sensorID = -1);

    bool begin(void);
    void setMagGain(lsm303MagGain gain);
    void getEvent(sensors_event_t*);
    void getSensor(sensor_t*);

  private:
    lsm303MagGain   _magGain;
    lsm303MagData   _magData;     // Last read magnetometer data will be available
here
    int32_t         _sensorID;

    void write8(byte address, byte reg, byte value);
    byte read8(byte address, byte reg);
    void read(void);
};
```

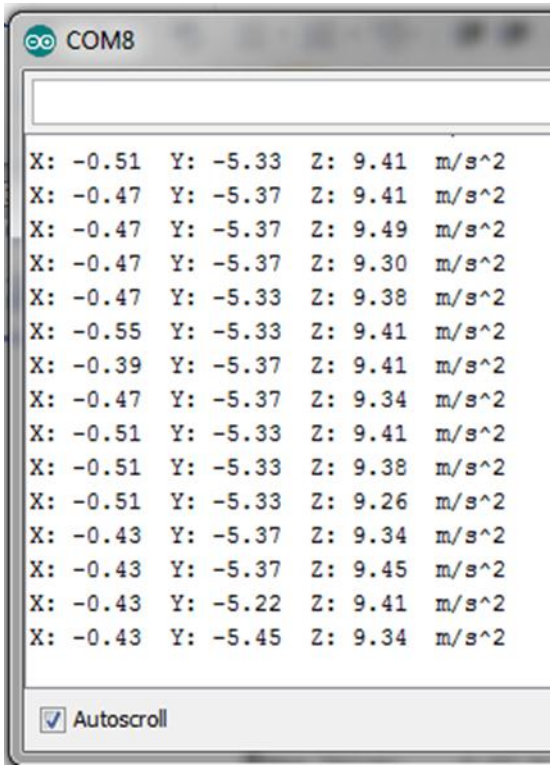
LSM303_U.cpp

```
/**************************************************************************/
/*!
    @brief  Sets the magnetometer's gain
*/
/**************************************************************************/
void LSM303_Mag_Unified::setMagGain(lsm303MagGain gain)
{
  write8(LSM303_ADDRESS_MAG, LSM303_REGISTER_MAG_CRB_REG_M, (byte)gain);

  _magGain = gain;

  switch(gain)
  {
    case LSM303_MAGGAIN_1_3:
      _lsm303Mag_Gauss_LSB_XY = 1100;
      _lsm303Mag_Gauss_LSB_Z  = 980;
      break;
    case LSM303_MAGGAIN_1_9:
      _lsm303Mag_Gauss_LSB_XY = 855;
      _lsm303Mag_Gauss_LSB_Z  = 760;
      break;
    case LSM303_MAGGAIN_2_5:
      _lsm303Mag_Gauss_LSB_XY = 670;
      _lsm303Mag_Gauss_LSB_Z  = 600;
      break;
    case LSM303_MAGGAIN_4_0:
      _lsm303Mag_Gauss_LSB_XY = 450;
      _lsm303Mag_Gauss_LSB_Z  = 400;
      break;
    case LSM303_MAGGAIN_4_7:
      _lsm303Mag_Gauss_LSB_XY = 400;
      _lsm303Mag_Gauss_LSB_Z  = 255;
      break;
    case LSM303_MAGGAIN_5_6:
      _lsm303Mag_Gauss_LSB_XY = 330;
      _lsm303Mag_Gauss_LSB_Z  = 295;
      break;
    case LSM303_MAGGAIN_8_1:
      _lsm303Mag_Gauss_LSB_XY = 230;
      _lsm303Mag_Gauss_LSB_Z  = 205;
      break;
  }
}

/**************************************************************************/
/*!
    @brief  Gets the most recent sensor event
*/
/**************************************************************************/
void LSM303_Mag_Unified::getEvent(sensors_event_t *event) {
  /* Clear the event */
  memset(event, 0, sizeof(sensors_event_t));

  /* Read new data */
  read();

  event->version   = sizeof(sensors_event_t);
  event->sensor_id = _sensorID;
  event->type      = SENSOR_TYPE_MAGNETIC_FIELD;
  event->timestamp = 0;
  event->magnetic.x = _magData.x / _lsm303Mag_Gauss_LSB_XY *
SENSORS_GAUSS_TO_MICROTESLA;
```

```
  event->magnetic.y = _magData.y / _lsm303Mag_Gauss_LSB_XY *
SENSORS_GAUSS_TO_MICROTESLA;
  event->magnetic.z = _magData.z / _lsm303Mag_Gauss_LSB_Z *
SENSORS_GAUSS_TO_MICROTESLA;
}

/***************************************************************************/
/*!
    @brief  Gets the sensor_t data
*/
/***************************************************************************/
void LSM303_Mag_Unified::getSensor(sensor_t *sensor) {
  /* Clear the sensor_t object */
  memset(sensor, 0, sizeof(sensor_t));

  /* Insert the sensor name in the fixed length char array */
  strncpy (sensor->name, "LSM303", sizeof(sensor->name) - 1);
  sensor->name[sizeof(sensor->name)- 1] = 0;
  sensor->version     = 1;
  sensor->sensor_id   = _sensorID;
  sensor->type        = SENSOR_TYPE_MAGNETIC_FIELD;
  sensor->min_delay   = 0;
  sensor->max_value   = 0.0F; // TBD
  sensor->min_value   = 0.0F; // TBD
  sensor->resolution  = 0.0F; // TBD
}
```

Výstup tohoto programu přes sériový port na monitoru je např.



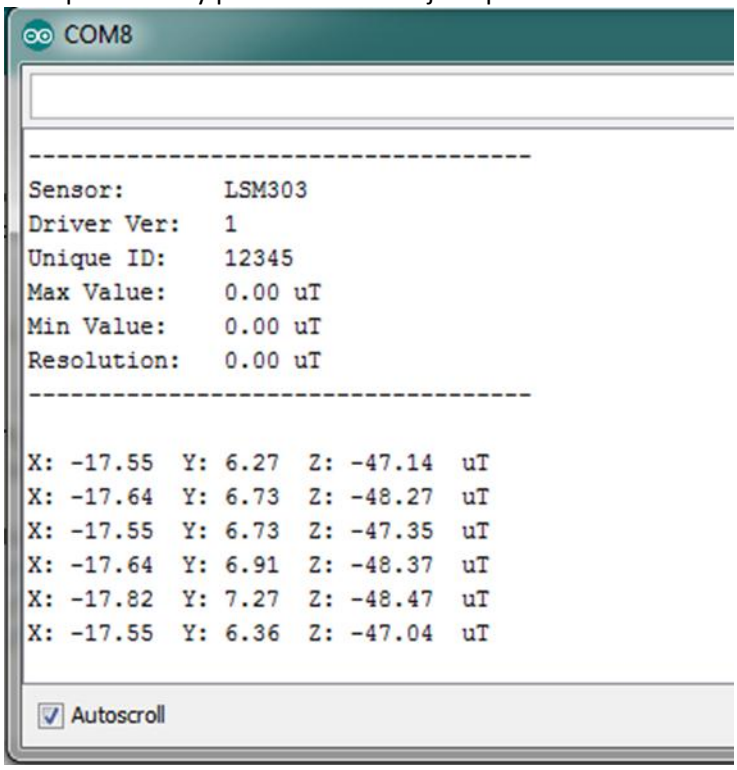


Měření magnetického pole.
Pro programovou obsluhu jsem odzkoušel následující sketch:

```
#include <Wire.h>
#include <Sensor.h>
#include <LSM303_U.h>

/* Assign a unique ID to this sensor at the same time */
LSM303_Mag_Unified mag = LSM303_Mag_Unified(12345);

void displaySensorDetails(void)
{
  sensor_t sensor;
  mag.getSensor(&sensor);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println("
uT");
  Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println("
uT");
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println("
uT");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

void setup(void)
{
  Serial.begin(9600);
  Serial.println("Magnetometer Test"); Serial.println("");
```

```
  /* Initialise the sensor */
  if(!mag.begin())
  {
    /* There was a problem detecting the LSM303 ... check your connections */
    Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
    while(1);
  }

  /* Display some basic information on this sensor */
  displaySensorDetails();
}

void loop(void)
{
  /* Get a new sensor event */
  sensors_event_t event;
  mag.getEvent(&event);

  /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
  Serial.print("X: "); Serial.print(event.magnetic.x); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event.magnetic.y); Serial.print("  ");
  Serial.print("Z: "); Serial.print(event.magnetic.z); Serial.print("
");Serial.println("uT");
  delay(500);
}
```

Potřebujeme k němu ještě knihovny Sensor a LSM303_U, stejné jako v předchozím odstavci (akcelerometr).
Výstup tohoto programu přes sériový port na monitoru je např.



Gyroskop.

Pro programovou obsluhu jsem odzkoušel následující sketch:

```
#include <Wire.h>
#include <L3G.h>

L3G gyro;

void setup() {
  Serial.begin(9600);
  Wire.begin();

  if (!gyro.init())
  {
    Serial.println("Failed to autodetect gyro type!");
    while (1);
  }

  gyro.enableDefault();
}

void loop() {
  gyro.read();

  Serial.print("G ");
  Serial.print("X: ");
  Serial.print((int)gyro.g.x);
  Serial.print(" Y: ");
  Serial.print((int)gyro.g.y);
  Serial.print(" Z: ");
  Serial.println((int)gyro.g.z);

  delay(100);
}
```

Potřebujeme k němu ještě knihovnu L3G:

L3G.h

```
#ifndef L3G_h
#define L3G_h

#include <Arduino.h> // for byte data type

// device types

#define L3G_DEVICE_AUTO 0
#define L3G4200D_DEVICE 1
#define L3GD20_DEVICE   2


// SA0 states

#define L3G_SA0_LOW  0
#define L3G_SA0_HIGH 1
#define L3G_SA0_AUTO 2

// register addresses

#define L3G_WHO_AM_I      0x0F

#define L3G_CTRL_REG1     0x20
```

```
#define L3G_CTRL_REG2     0x21
#define L3G_CTRL_REG3     0x22
#define L3G_CTRL_REG4     0x23
#define L3G_CTRL_REG5     0x24
#define L3G_REFERENCE     0x25
#define L3G_OUT_TEMP      0x26
#define L3G_STATUS_REG    0x27

#define L3G_OUT_X_L       0x28
#define L3G_OUT_X_H       0x29
#define L3G_OUT_Y_L       0x2A
#define L3G_OUT_Y_H       0x2B
#define L3G_OUT_Z_L       0x2C
#define L3G_OUT_Z_H       0x2D

#define L3G_FIFO_CTRL_REG 0x2E
#define L3G_FIFO_SRC_REG  0x2F

#define L3G_INT1_CFG      0x30
#define L3G_INT1_SRC      0x31
#define L3G_INT1_THS_XH   0x32
#define L3G_INT1_THS_XL   0x33
#define L3G_INT1_THS_YH   0x34
#define L3G_INT1_THS_YL   0x35
#define L3G_INT1_THS_ZH   0x36
#define L3G_INT1_THS_ZL   0x37
#define L3G_INT1_DURATION 0x38

class L3G
{
  public:
    typedef struct vector
    {
      float x, y, z;
    } vector;

    vector g; // gyro angular velocity readings

    bool init(byte device = L3G_DEVICE_AUTO, byte sa0 = L3G_SA0_AUTO);

    void enableDefault(void);

    void writeReg(byte reg, byte value);
    byte readReg(byte reg);

    void read(void);

    // vector functions
    static void vector_cross(const vector *a, const vector *b, vector *out);
    static float vector_dot(const vector *a,const vector *b);
    static void vector_normalize(vector *a);

  private:
      byte _device; // chip type (4200D or D20)
      byte address;

      bool autoDetectAddress(void);
};

#endif
```

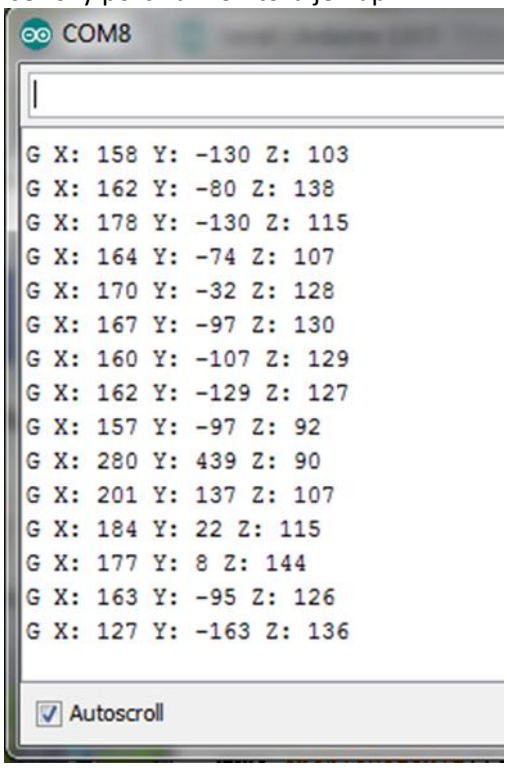
L3G.cpp

```
#include <L3G.h>
#include <Wire.h>
#include <math.h>

// Defines /////////////////////////////////////////////////////////////////

// The Arduino two-wire interface uses a 7-bit number for the address,
// and sets the last bit correctly based on reads and writes
#define L3G4200D_ADDRESS_SA0_LOW  (0xD0 >> 1)
#define L3G4200D_ADDRESS_SA0_HIGH (0xD2 >> 1)
#define L3GD20_ADDRESS_SA0_LOW    (0xD4 >> 1)
#define L3GD20_ADDRESS_SA0_HIGH   (0xD6 >> 1)

// Public Methods //////////////////////////////////////////////////////////////

bool L3G::init(byte device, byte sa0)
{
  _device = device;
  switch (_device)
  {
    case L3G4200D_DEVICE:
      if (sa0 == L3G_SA0_LOW)
      {
        address = L3G4200D_ADDRESS_SA0_LOW;
        return true;
      }
      else if (sa0 == L3G_SA0_HIGH)
      {
        address = L3G4200D_ADDRESS_SA0_HIGH;
        return true;
      }
      else
        return autoDetectAddress();
      break;

    case L3GD20_DEVICE:
      if (sa0 == L3G_SA0_LOW)
      {
        address = L3GD20_ADDRESS_SA0_LOW;
        return true;
      }
      else if (sa0 == L3G_SA0_HIGH)
      {
        address = L3GD20_ADDRESS_SA0_HIGH;
        return true;
      }
      else
        return autoDetectAddress();
      break;

    default:
      return autoDetectAddress();
  }
}

// Turns on the L3G's gyro and places it in normal mode.
void L3G::enableDefault(void)
{
  // 0x0F = 0b00001111
  // Normal power mode, all axes enabled
```

```
  writeReg(L3G_CTRL_REG1, 0x0F);
}

// Writes a gyro register
void L3G::writeReg(byte reg, byte value)
{
  Wire.beginTransmission(address);
  Wire.write(reg);
  Wire.write(value);
  Wire.endTransmission();
}

// Reads a gyro register
byte L3G::readReg(byte reg)
{
  byte value;

  Wire.beginTransmission(address);
  Wire.write(reg);
  Wire.endTransmission();
  Wire.requestFrom(address, (byte)1);
  value = Wire.read();
  Wire.endTransmission();

  return value;
}

// Reads the 3 gyro channels and stores them in vector g
void L3G::read()
{
  Wire.beginTransmission(address);
  // assert the MSB of the address to get the gyro
  // to do slave-transmit subaddress updating.
  Wire.write(L3G_OUT_X_L | (1 << 7));
  Wire.endTransmission();
  Wire.requestFrom(address, (byte)6);

  while (Wire.available() < 6);

  uint8_t xlg = Wire.read();
  uint8_t xhg = Wire.read();
  uint8_t ylg = Wire.read();
  uint8_t yhg = Wire.read();
  uint8_t zlg = Wire.read();
  uint8_t zhg = Wire.read();

  // combine high and low bytes
  g.x = (int16_t)(xhg << 8 | xlg);
  g.y = (int16_t)(yhg << 8 | ylg);
  g.z = (int16_t)(zhg << 8 | zlg);
}

void L3G::vector_cross(const vector *a,const vector *b, vector *out)
{
  out->x = a->y*b->z - a->z*b->y;
  out->y = a->z*b->x - a->x*b->z;
  out->z = a->x*b->y - a->y*b->x;
}

float L3G::vector_dot(const vector *a,const vector *b)
{
  return a->x*b->x+a->y*b->y+a->z*b->z;
```

```
}

void L3G::vector_normalize(vector *a)
{
  float mag = sqrt(vector_dot(a,a));
  a->x /= mag;
  a->y /= mag;
  a->z /= mag;
}

// Private Methods //////////////////////////////////////////////////////////////

bool L3G::autoDetectAddress(void)
{
  // try each possible address and stop if reading WHO_AM_I returns the expected
response
  address = L3G4200D_ADDRESS_SA0_LOW;
  if (readReg(L3G_WHO_AM_I) == 0xD3) return true;
  address = L3G4200D_ADDRESS_SA0_HIGH;
  if (readReg(L3G_WHO_AM_I) == 0xD3) return true;
  address = L3GD20_ADDRESS_SA0_LOW;
  if (readReg(L3G_WHO_AM_I) == 0xD4) return true;
  address = L3GD20_ADDRESS_SA0_HIGH;
  if (readReg(L3G_WHO_AM_I) == 0xD4) return true;

  return false;
}
```

Výstup tohoto programu přes sériový port na monitoru je např.



Měření atmosférického tlaku.

Pro programovou obsluhu jsem odzkoušel následující sketch:

```
#include <Wire.h>
#include <LPS331.h>

LPS331 ps;

void setup()
{
  Serial.begin(9600);
  Wire.begin();

  if (!ps.init())
  {
    Serial.println("Failed to autodetect pressure sensor!");
    while (1);
  }

  ps.enableDefault();
}

void loop()
{
  float pressure = ps.readPressureMillibars();
  float altitude = ps.pressureToAltitudeMeters(pressure);
  float temperature = ps.readTemperatureC();

  Serial.print("p: ");
  Serial.print(pressure);
  Serial.print(" mbar\ta: ");
  Serial.print(altitude);
  Serial.print(" m\tt: ");
  Serial.print(temperature);
  Serial.println(" deg C");

  delay(1000);
}
```

Potřebujeme k němu ještě knihovnu LPS331:

LPS331.h

```
#ifndef LPS331_h
#define LPS331_h

#include <Arduino.h> // for byte data type

// SA0 states

#define LPS331_SA0_LOW  0
#define LPS331_SA0_HIGH 1
#define LPS331_SA0_AUTO 2

// register addresses
// Note: Some of the register names in the datasheet are inconsistent
// between Table 14 in section 6 and the register descriptions in
// section 7. Where they differ, the names from section 7 have been
// used here.

#define LPS331_REF_P_XL       0x08
#define LPS331_REF_P_L        0x09
```

```
#define LPS331_REF_P_H        0x0A

#define LPS331_WHO_AM_I       0x0F

#define LPS331_RES_CONF       0x10

#define LPS331_CTRL_REG1      0x20
#define LPS331_CTRL_REG2      0x21
#define LPS331_CTRL_REG3      0x22
#define LPS331_INTERRUPT_CFG  0x23
#define LPS331_INT_SOURCE     0x24
#define LPS331_THS_P_L        0x25
#define LPS331_THS_P_H        0x26
#define LPS331_STATUS_REG     0x27

#define LPS331_PRESS_OUT_XL   0x28
#define LPS331_PRESS_OUT_L    0x29
#define LPS331_PRESS_OUT_H    0x2A

#define LPS331_TEMP_OUT_L     0x2B
#define LPS331_TEMP_OUT_H     0x2C

#define LPS331_AMP_CTRL       0x30

#define LPS331_DELTA_PRESS_XL 0x3C
#define LPS331_DELTA_PRESS_L  0x3D
#define LPS331_DELTA_PRESS_H  0x3E

class LPS331
{
  public:
    LPS331(void);

    bool init(byte sa0 = LPS331_SA0_AUTO);

    void enableDefault(void);

    void writeReg(byte reg, byte value);
    byte readReg(byte reg);

    float readPressureMillibars(void);
    float readPressureInchesHg(void);
    int32_t readPressureRaw(void);
    float readTemperatureC(void);
    float readTemperatureF(void);
    int16_t readTemperatureRaw(void);

    static float pressureToAltitudeMeters(float pressure_mbar, float
altimeter_setting_mbar = 1013.25);
    static float pressureToAltitudeFeet(float pressure_inHg, float
altimeter_setting_inHg = 29.9213);

  private:
    byte address;

    bool autoDetectAddress(void);
    bool testWhoAmI(void);
};

#endif
```

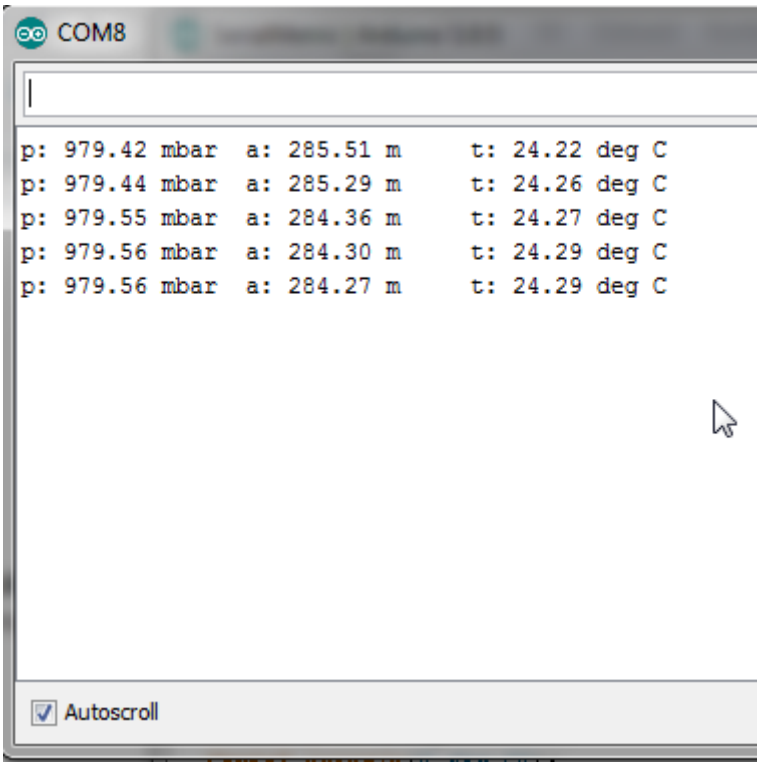
LPS331.cpp

```
#include <LPS331.h>
#include <Wire.h>

// Defines /////////////////////////////////////////////////////////////

// The Arduino two-wire interface uses a 7-bit number for the address,
// and sets the last bit correctly based on reads and writes
#define LPS331AP_ADDRESS_SA0_LOW  0b1011100
#define LPS331AP_ADDRESS_SA0_HIGH 0b1011101

// Constructors ////////////////////////////////////////////////////////

LPS331::LPS331(void)
{
  // Pololu board pulls SA0 high, so default assumption is that it is
  // high
  address = LPS331AP_ADDRESS_SA0_HIGH;
}

// Public Methods //////////////////////////////////////////////////////

// sets or detects slave address; returns bool indicating success
bool LPS331::init(byte sa0)
{
  switch(sa0)
  {
    case LPS331_SA0_LOW:
      address = LPS331AP_ADDRESS_SA0_LOW;
      return testWhoAmI();

    case LPS331_SA0_HIGH:
      address = LPS331AP_ADDRESS_SA0_HIGH;
      return testWhoAmI();

    default:
      return autoDetectAddress();
  }
}

// turns on sensor and enables continuous output
void LPS331::enableDefault(void)
{
  // active mode, 12.5 Hz output data rate
  writeReg(LPS331_CTRL_REG1, 0b11100000);
}

// writes register
void LPS331::writeReg(byte reg, byte value)
{
  Wire.beginTransmission(address);
  Wire.write(reg);
  Wire.write(value);
  Wire.endTransmission();
}

// reads register
byte LPS331::readReg(byte reg)
```

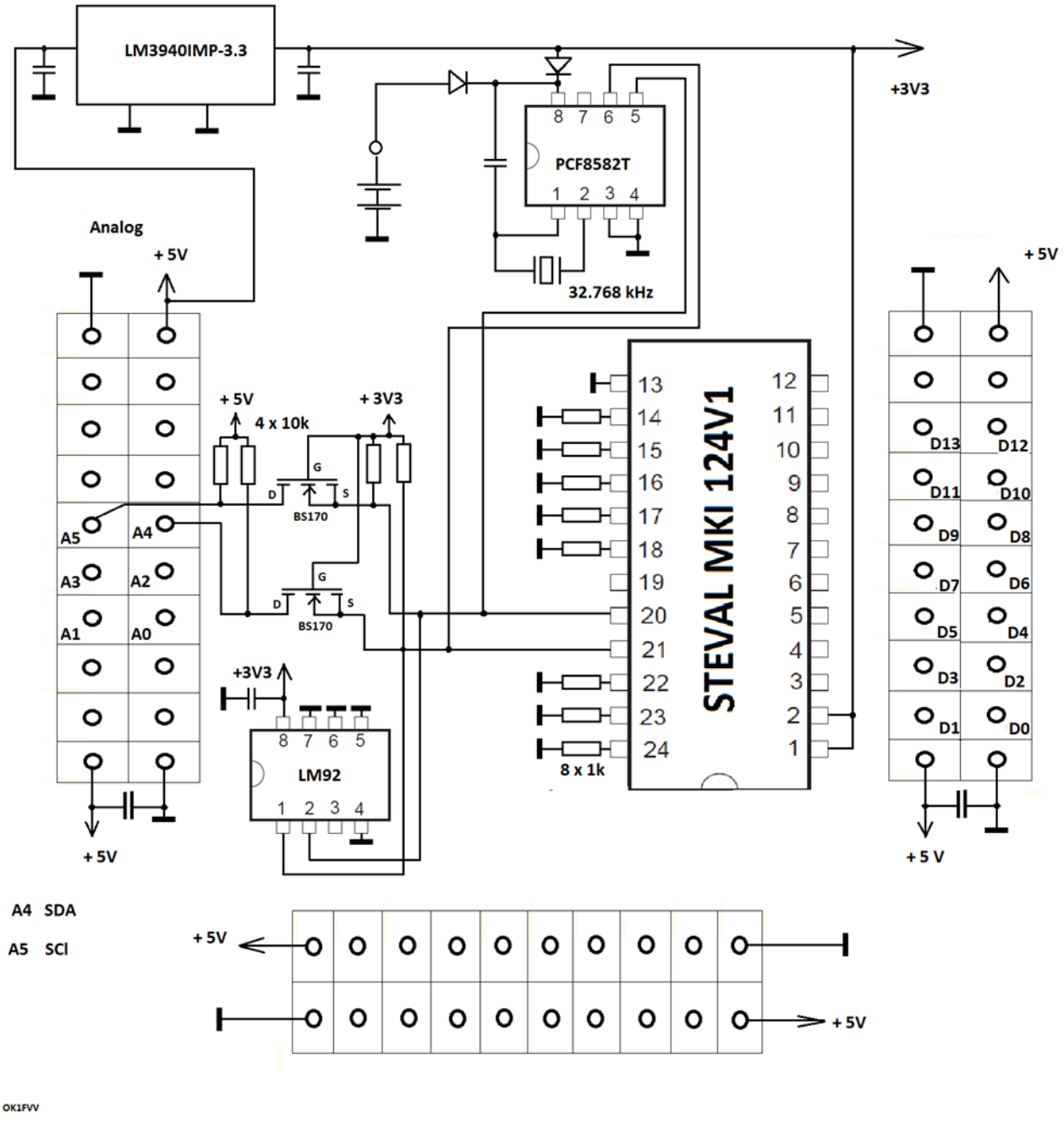
```
{
  byte value;

  Wire.beginTransmission(address);
  Wire.write(reg);
  Wire.endTransmission(false); // restart
  Wire.requestFrom(address, (byte)1);
  value = Wire.read();
  Wire.endTransmission();

  return value;
}

// reads pressure in millibars (mbar)/hectopascals (hPa)
float LPS331::readPressureMillibars(void)
{
  return (float)readPressureRaw() / 4096;
}

// reads pressure in inches of mercury (inHg)
float LPS331::readPressureInchesHg(void)
{
  return (float)readPressureRaw() / 138706.5;
}

// reads pressure and returns raw 24-bit sensor output
int32_t LPS331::readPressureRaw(void)
{
  Wire.beginTransmission(address);
  // assert MSB to enable register address auto-increment
  Wire.write(LPS331_PRESS_OUT_XL | (1 << 7));
  Wire.endTransmission();
  Wire.requestFrom(address, (byte)3);

  while (Wire.available() < 3);

  uint8_t pxl = Wire.read();
  uint8_t pl = Wire.read();
  uint8_t ph = Wire.read();

  // combine bytes
  return (int32_t)(int8_t)ph << 16 | (uint16_t)pl << 8 | pxl;
}

// reads temperature in degrees C
float LPS331::readTemperatureC(void)
{
  return 42.5 + (float)readTemperatureRaw() / 480;
}

// reads temperature in degrees F
float LPS331::readTemperatureF(void)
{
  return 108.5 + (float)readTemperatureRaw() / 480 * 1.8;
}

// reads temperature and returns raw 16-bit sensor output
int16_t LPS331::readTemperatureRaw(void)
{
  Wire.beginTransmission(address);
  // assert MSB to enable register address auto-increment
  Wire.write(LPS331_TEMP_OUT_L | (1 << 7));
```

```
  Wire.endTransmission();
  Wire.requestFrom(address, (byte)2);

  while (Wire.available() < 2);

  uint8_t tl = Wire.read();
  uint8_t th = Wire.read();

  // combine bytes
  return (int16_t)(th << 8 | tl);
}

// converts pressure in mbar to altitude in meters, using 1976 US
// Standard Atmosphere model (note that this formula only applies to a
// height of 11 km, or about 36000 ft)
//  If altimeter setting (QNH, barometric pressure adjusted to sea
//  level) is given, this function returns an indicated altitude
//  compensated for actual regional pressure; otherwise, it returns
//  the pressure altitude above the standard pressure level of 1013.25
//  mbar or 29.9213 inHg
float LPS331::pressureToAltitudeMeters(float pressure_mbar, float
altimeter_setting_mbar)
{
  return (1 - pow(pressure_mbar / altimeter_setting_mbar, 0.190263)) * 44330.8;
}

// converts pressure in inHg to altitude in feet; see notes above
float LPS331::pressureToAltitudeFeet(float pressure_inHg, float
altimeter_setting_inHg)
{
  return (1 - pow(pressure_inHg / altimeter_setting_inHg, 0.190263)) * 145442;
}

// Private Methods //////////////////////////////////////////////////

bool LPS331::autoDetectAddress(void)
{
  // try each possible address and stop if reading WHO_AM_I returns the expected
response
  address = LPS331AP_ADDRESS_SA0_LOW;
  if (testWhoAmI()) return true;
  address = LPS331AP_ADDRESS_SA0_HIGH;
  if (testWhoAmI()) return true;

  return false;
}

bool LPS331::testWhoAmI(void)
{
  return (readReg(LPS331_WHO_AM_I) == 0xBB);
}
```

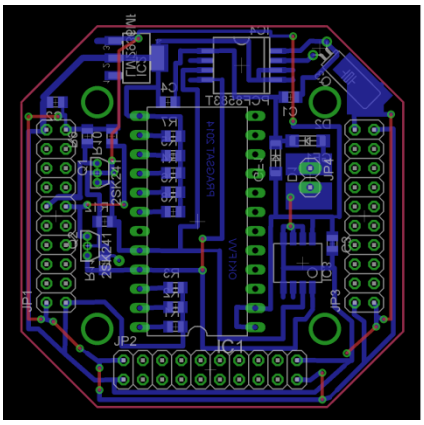
Výstup tohoto programu přes sériový port na monitoru je např.

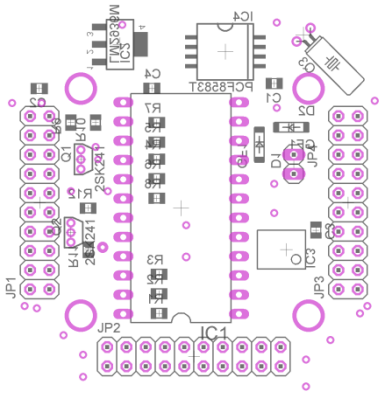


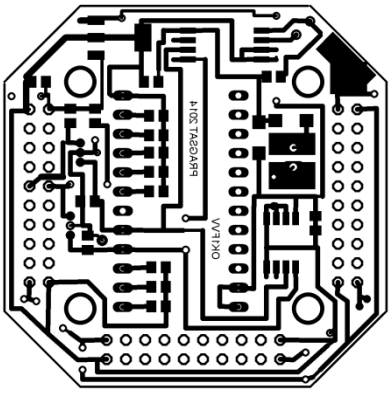
Pro slučitelnost s startkitem T-minus Engineering jsem vytvořil následující modul a program zpracovávající data z jeho čidel:

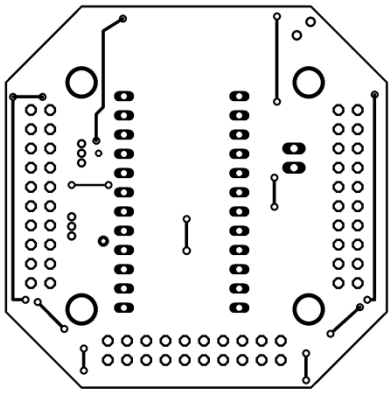


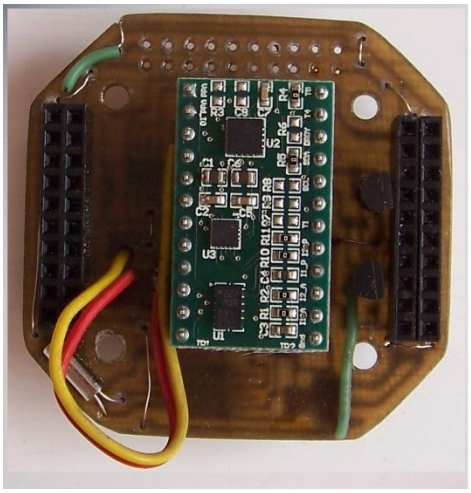
Na desce najdeme především destičku STEVAL MKI12V1 s čidly. Protože destička čidel má napájení 3V3, je na naší desce ještě zdroj 3V3 tvořený LM3940IMP-3.3 a dále dvojice tranzistorů BS170 zapojená jako obousměrné převodníky úrovní 5V – 3V3 pro signály SDA a SCL sběrmice I2C. Na destičce ještě najdeme I2C čidlo teploty LM92 (přesnmost 0,33 stupně) a I2C obvod hodin PCF8582T.

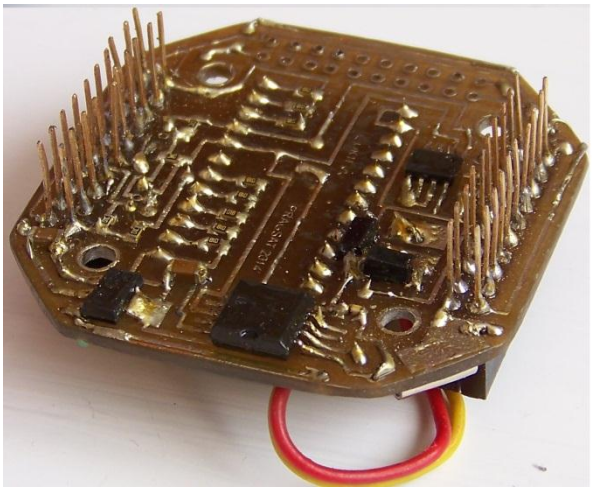












Program pro čidla na této desce:

```
#include <Wire.h> // necessary, or the application won't build properly
#include <stdio.h>
#include <PCF8583.h>
#include <LPS331.h>
#include <Sensor.h>
#include <LSM303_U.h>
```

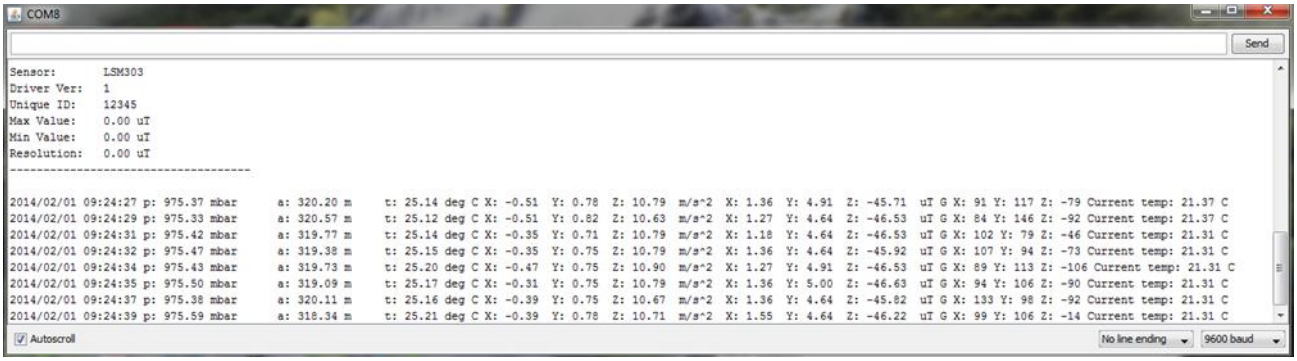
```
#include <L3G.h>
#include <LM76.h>

/**************************************************************************
 *  all CanSAT CZ sensors
 *  Pragsat 2014   ver. 0.1   31.1.2014
 **************************************************************************/
LM76 sensor;

LPS331 ps;

int correct_address = 0;
PCF8583 p (0xA0);

/* Assign a unique ID to this sensor at the same time */
LSM303_Accel_Unified accel = LSM303_Accel_Unified(54321);

/* Assign a unique ID to this sensor at the same time */
LSM303_Mag_Unified mag = LSM303_Mag_Unified(12345);

L3G gyro;

 void displaySensorDetails(void)
{
  sensor_t sensor1;
  accel.getSensor(&sensor1);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor1.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor1.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor1.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor1.max_value); Serial.println("
m/s^2");
  Serial.print  ("Min Value:    "); Serial.print(sensor1.min_value); Serial.println("
m/s^2");
  Serial.print  ("Resolution:   "); Serial.print(sensor1.resolution); Serial.println("
m/s^2");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);

  sensor_t sensor2;
  mag.getSensor(&sensor2);
  Serial.println("------------------------------------");
  Serial.print  ("Sensor:       "); Serial.println(sensor2.name);
  Serial.print  ("Driver Ver:   "); Serial.println(sensor2.version);
  Serial.print  ("Unique ID:    "); Serial.println(sensor2.sensor_id);
  Serial.print  ("Max Value:    "); Serial.print(sensor2.max_value); Serial.println("
uT");
  Serial.print  ("Min Value:    "); Serial.print(sensor2.min_value); Serial.println("
uT");
  Serial.print  ("Resolution:   "); Serial.print(sensor2.resolution); Serial.println("
uT");
  Serial.println("------------------------------------");
  Serial.println("");
  delay(500);
}

void setup(void){
  Serial.begin(9600);
  Serial.print("booting...");
  Serial.println(" done");
//mereni tlaku
```

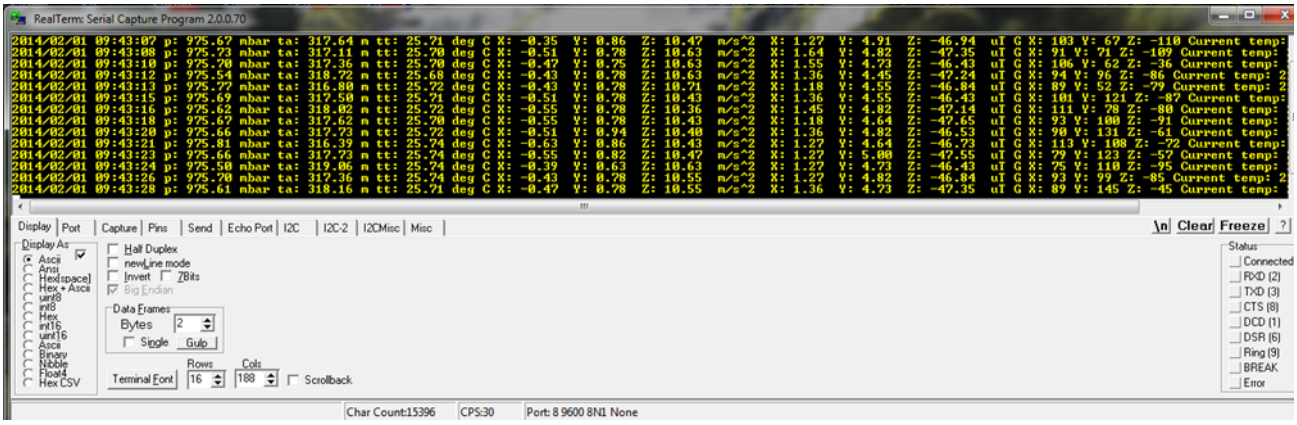
```
 Wire.begin();

  if (!ps.init())
  {
    Serial.println("Failed to autodetect pressure sensor!");
    while (1);
  }

  ps.enableDefault();
//akcelerometr
  Serial.println("Accelerometer Test"); Serial.println("");

  /* Initialise the sensor */
  if(!accel.begin())
  {
    /* There was a problem detecting the ADXL345 ... check your connections */
    Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
    while(1);
  }

  /* Display some basic information on this sensor */
  displaySensorDetails();
//magnetometr
Serial.println("Magnetometer Test"); Serial.println("");

  /* Initialise the sensor */
  if(!mag.begin())
  {
    /* There was a problem detecting the LSM303 ... check your connections */
    Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
    while(1);
  }

  /* Display some basic information on this sensor */
  displaySensorDetails();
//gyroskope
  if (!gyro.init())
  {
    Serial.println("Failed to autodetect gyro type!");
    while (1);
  }

  gyro.enableDefault();

}

void loop(void){
 if(Serial.available() > 0){
       p.year= (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48)) + 2000;
       p.month = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
       p.day = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
       p.hour  = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
       p.minute = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
       p.second = (byte) ((Serial.read() - 48) * 10 + (Serial.read() - 48)); // Use of
(byte) type casting and ascii math to achieve result.

       if(Serial.read() == ';'){
         Serial.println("setting date");
        p.set_time();
       }
  }
```

```
  p.get_time();
  char time[50];
  sprintf(time, "%02d/%02d/%02d %02d:%02d:%02d",
         p.year, p.month, p.day, p.hour, p.minute, p.second);
  Serial.print(time);Serial.print(" ");
  // mereni tlaku - zacatek
   float pressure = ps.readPressureMillibars();
  float altitude = ps.pressureToAltitudeMeters(pressure);
  float temperature = ps.readTemperatureC();

  Serial.print("p: ");
  Serial.print(pressure);
  //Serial.print(" mbar\ta: ");
  Serial.print(" mbar ta: ");
  Serial.print(altitude);
  //Serial.print(" m\tt: ");
  Serial.print(" m tt: ");
  Serial.print(temperature);
  Serial.print(" deg C");
  Serial.print(" ");
  //akcelerometr
   /* Get a new sensor event */
  sensors_event_t event1;
  accel.getEvent(&event1);

  /* Display the results (acceleration is measured in m/s^2) */
  Serial.print("X: "); Serial.print(event1.acceleration.x); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event1.acceleration.y); Serial.print("  ");
  Serial.print("Z: "); Serial.print(event1.acceleration.z); Serial.print("  ");
  Serial.print("m/s^2 ");
  Serial.print(" ");
  //magnetometr
  /* Get a new sensor event */
  sensors_event_t event;
  mag.getEvent(&event);

  /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
  Serial.print("X: "); Serial.print(event.magnetic.x); Serial.print("  ");
  Serial.print("Y: "); Serial.print(event.magnetic.y); Serial.print("  ");
  Serial.print("Z: "); Serial.print(event.magnetic.z); Serial.print("  ");
  Serial.print("uT");
  Serial.print(" ");
   //gyroskope
   gyro.read();

  Serial.print("G ");
  Serial.print("X: ");
  Serial.print((int)gyro.g.x);
  Serial.print(" Y: ");
  Serial.print((int)gyro.g.y);
  Serial.print(" Z: ");
  Serial.print((int)gyro.g.z);
  Serial.print(" ");
   //teplomer LM76
// get temperature from sensor
  Serial.print("Current temp: ");
  Serial.print(sensor.temp());
  Serial.print(" C");
  Serial.print(" ");
  sensor.shutdown(true);
  delay(1000);
```

```
  // wake up sensor for next time around
  sensor.shutdown(false);
   //////
  Serial.println(" ");
  delay(500);
}
```

Výstup tohoto programu přes sériový port na monitoru je např.





4. Komunikační systém

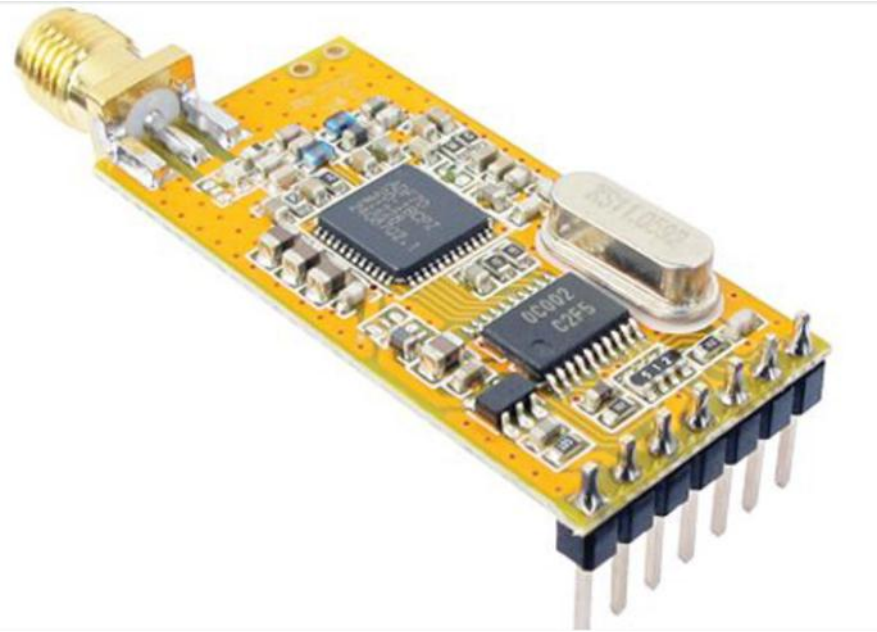
V prvních dvou ročnících soutěže byl používán startkit od Pratt Hobies s vysílačem s ADF7012 pro pásmo 70cm (433 MHz). Počínajíc třetím ročníkem jsou používány trancievery (vysílač a přijímač) pro možnost oboustranného spojenímezi CanSATem a pozemní stanicí. Nicméně ve většině případů je stejně využíván přenos dat z CanSATu do pozemní stanice.

Ve třetím ročníku byly používány trancievery APC220 čínské výroby. Ve čtvrtém ročníku máme k dispozici jak tento tcvr, tak nově i tcvr od T-minus Engineering.

4.1 TCVR APC22O

Ve třetím ročníku soutěže European Cansat Competition 2013 byly používány trancievery APC220. Nikde jsem ovšem nenašel zapojení trancieveru ani firmware jeho mikrořadiče a tak jsem se pokusil o jeho analýzu. Je z ní zřejmé, že vůbec chybí antenní filtr a lze tedy

předpokládat nevalné vlastnosti tcvr pokud jde o elektromagnetickou kompatibilitu (tj tcvr možná produkuje nekvalitní vf signál). Proto jsem asi také nenašel žádné certifikace ohledně elektromagnetické slučitelnosti udělené FCC či EU orgány.
Na druhé straně je pravda, že můžeme tcvr doplnit o vlastní antenní filtr a tím jeho vlastnosti zlepšit
Tranciever RF7020, jehož základem je obvod vysílače/přijímač ADF7020 firmy Analog Devices vyrábí pod názvem APC220 čínská firma SHENZHEN APPCON TECHNOLOGIES CO.LTD (www.appcon.com.cn) a pod názvem DRF7020D20 firma DORJI Applied Techmologies.(www.dorji.com) Tranciever se vyznačuje malými rozměry 37.5mm x 18.3mm x 7.0mm. Z jeho obrázku je zřejmé jeho konstrukční provedení:



Výrobci uvádějí následující parametry trancieveru:
Frekvenční rozsah 418MHz až 455MHz s krokem 1kHz, kanály po 200kHz
Výstupní výkon regulovatelný v 10 stupních , max. 20mW
GFSK modulace s modulační odchylkou 28,8kHz
UART interface s úrovní signálů TTL a rychlostí 1200,2400,4800,9600b,19200,38400,57600
Rychlost vysílání na vf 2400bps,4800bps,9600bps,19200bps
Napájecí napětí 3,4 až 5,5 V

Zapojení konektoru:

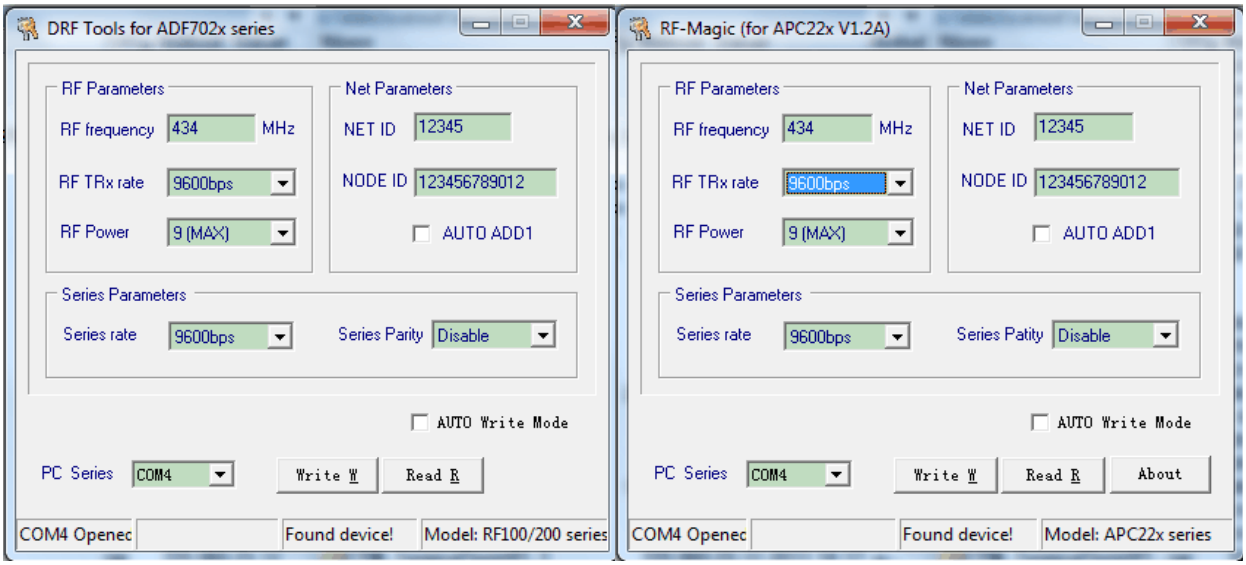
pin	jméno	funkce	
1	GND	Společná země	GND
2	VCC	napájení	napájení
3	EN	enable	vstup
4	RxD	Vstup UART	vstup
5	TxD	Výstup UART	výstup
6	AUX	Indikace data in/data out	výstup
7	SET	Nastavení, aktivní při L	vstup

Nastavování a komunikace PC s TCVR

Obyčejně jsou dodávány dva trancievery (předpokládáme totiž oboustranné spojení) spolu s USB adapterem.

Vidíme ho i na obrázku:

Je to převodník mezi USB a UARTem na úrovni TTL od firmy DFRobot.com. Obsahuje obvod CP2012 firmy Silicon Laboratories z jejichž stránek
http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx
také stáhneme potřebné drivery.
Převodník má na jedné straně USB, na straně druhé signály TxD, RxD a RTS. Při jeho propojení s tcvr se signál TxD převodníku spojí s RxD trancieveru, RxD převodníku s TxD trancieveru a RTS převodníku s EN trancieveru. Dále jsou ještě propojeny GND a kladné napájecí napětí VCC. Piny 6 a 7 trancieveru nejsou s převodníkem propojeny. Pokud máme výše uvedený převodník propojen s tcvr, můžeme si přečíst konfiguraci tcvr či provést nové nastavení.
Pro nastavování TCVR můžeme použít program **RF-Magic (for APC22x v.1.2A)** nebo prakticky totožný **DRF Tools for ADF702x series**



Pokud je vše v pořádku, program nalezne tcvr (Found device) i jeho model. Musíme mít ovšem PC Series nastavený na virtuální sériový port odpovídající převodníku USB a rovněž přenosovou rychlost UARTu tcvr. Ta je defaultně 9600 Bd. Činnost programu je velice jednoduchá: Po kliknutí na tlačítko **Read** přečte program nastavení tcvr. Pokud naopak vyplníme textová políčka programu a klikneme na tlačítko **Write**, nakonfiguruje se těmito hodnotami tcvr.
Program mi nepracoval na 64 bitových WIN7. Na 32bitových WIN7 a na WIN XP byl funkční. V některých případech přestal pracovat a bylo nutné restartovat PC nebo alespoň ukončit tento program pomocí *Správce úloh Windows*. Především potřeboval výše uvedený převodník z USB s CP2012. Pokud jsem ale použil tento převodník jen se signály RxD a TxD , bez RTS , program nepoznal připojený tcvr. Odzkoušel jsem, že převodník s CP2102 lze sice použít jen se signály TxD a RxD, ale program rozpozná tcvr až poté, co pin 3 tcvr tj EN na chvíli připojíme na L a poté na chvíli na H. Od toho okamžiku je pak tcvr v konfiguračním režimu.

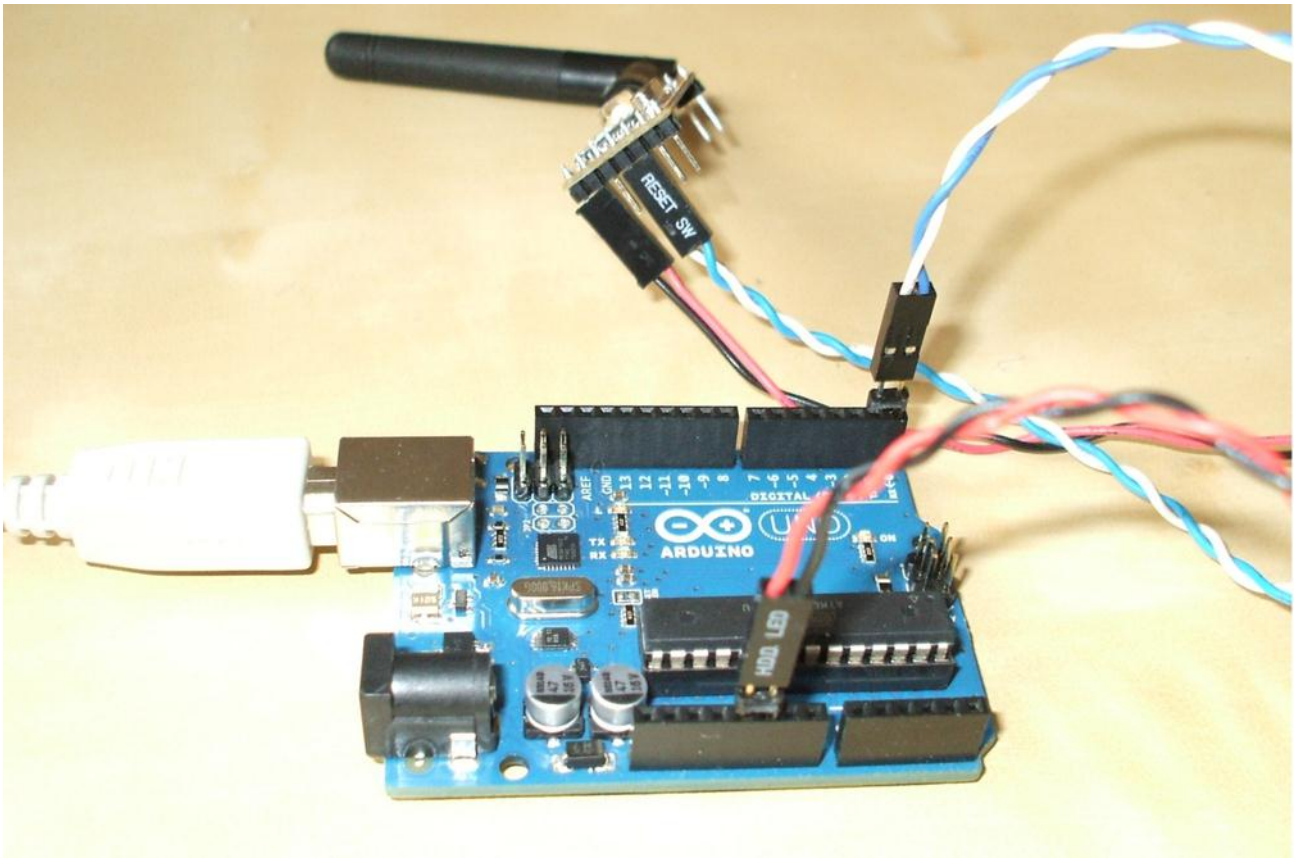
Nejdůležitější pro funkci tohoto programu je ale to, že musí být spuštěn pod administrátorským účtem.

Vysvětlení a popis této vlastnosti tcvr jsem v dostupné dokumentaci nenašel. Tam se naopak tvrdí, že do konfiguračního režimu se tcvr dostane úrovní L na pinu 7 (SET). Ten však není k adapteru USB připojen a program ho tedy nevyužívá na rozdíl od programu *arduina* uvedeného v tomto textu o tři strany dále.
Rozchodit program s USB převodníkem s FT232RL se mi nepodařilo, byť po připojení EN k L a poté k H chvilkami v patičce programu problikával nápis **Found device** a **Model:RF100/200 series** …
Dále provedeme ověření funkčnosti obou trancieverů. Využijeme přitom *Arduino Uno*, do něj uložíme následující kód:

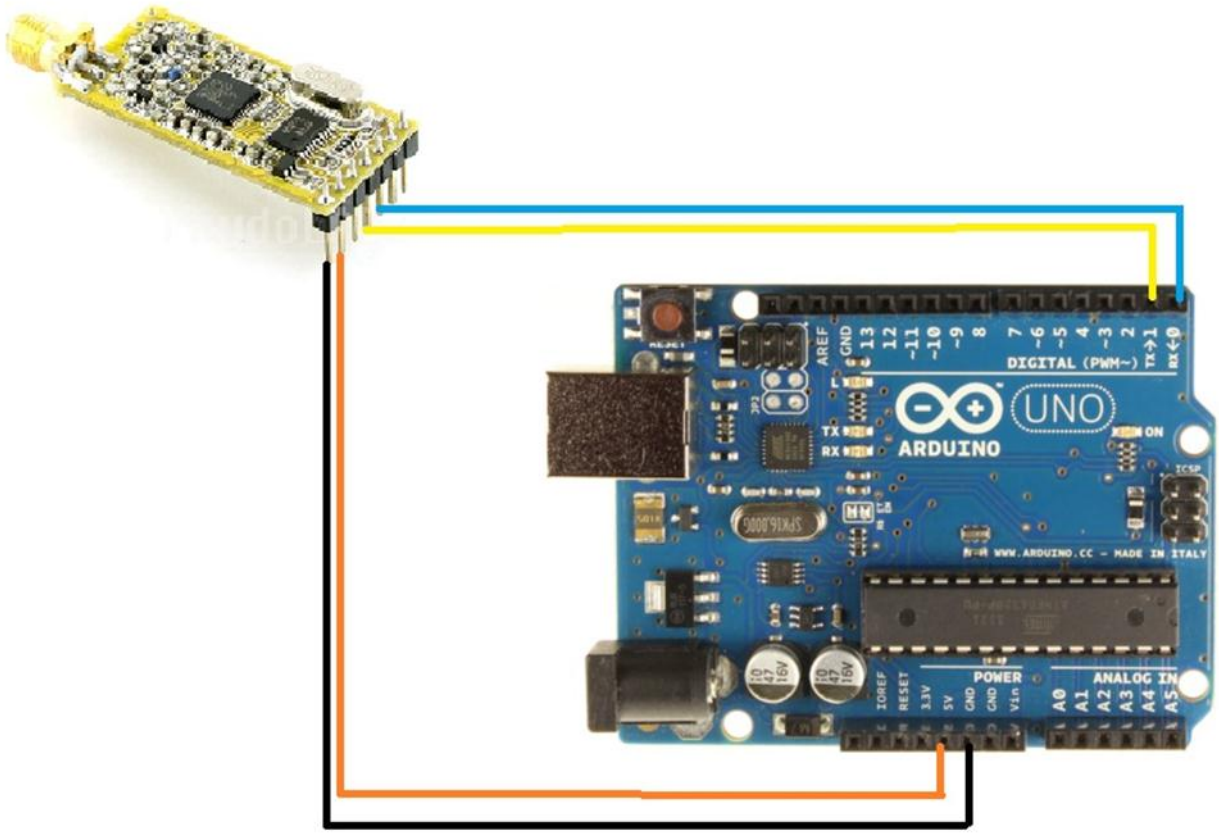
```
int val = 0;
int ledPin = 13;
void setup()
{
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop()
{
```

```
  val = Serial.read();
  if (-1 != val) {
   if ('A' == val || 'a' == val) {
     Serial.println("Arduino for CANSAT CZ");
   }else if ('B' == val || 'b' == val) {
      digitalWrite(ledPin, HIGH);
      delay(500);
      digitalWrite(ledPin, LOW);
    }
  }
}
```

K tomuto *Arduino Uno* připojíme jeden z trancieverů:



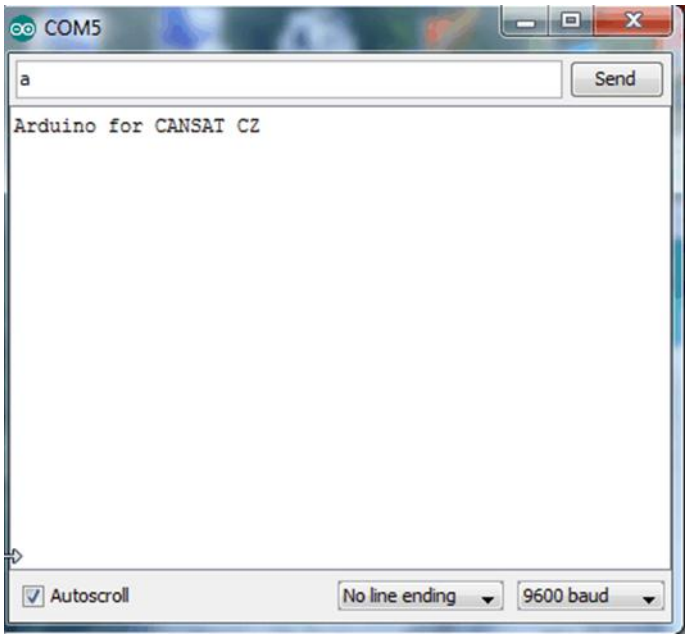
Všimněme si, že k připojení stačily čtyři vodiče: červeným (+5V) a černým (GND) jsme připojili k TCVR napájení z *arduina.* To je napájeno 5V přes USB kabel z PC. Dále propojíme RxD a TxD TCVR s TxD a RxD *arduina* (překříženě):



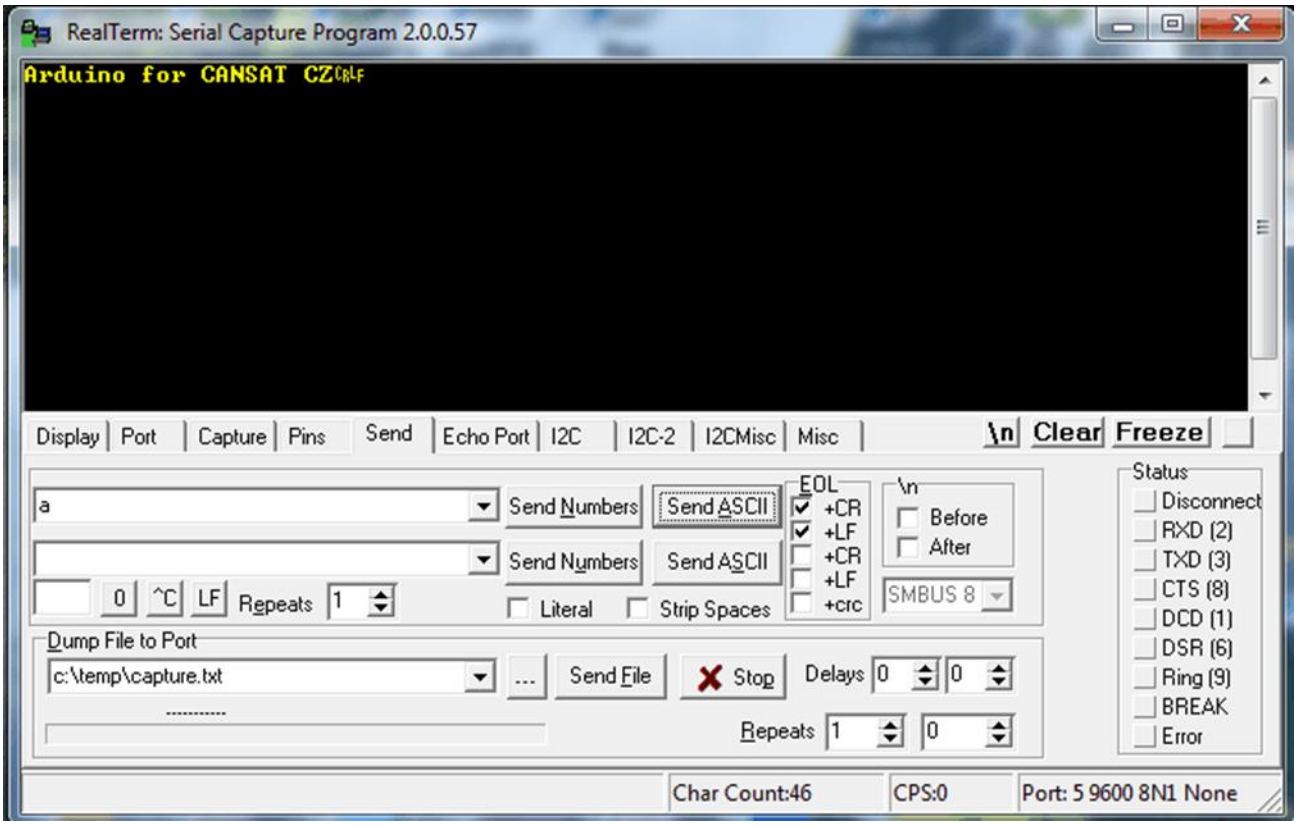
Druhý TCVR připojíme přes redukci USB – UART (TTL) (třeba tu s CP2012 od *DFRobot.com*, která se dodává s APC220) k PC. Na PC spustíme nějaký terminálový program. Nastavíme u něj přenosou rychlost 9600Bd, vysílání bez parity a to na virtuálním portu, k němuž přes USB máme připojen TCVR. Jestliže nyní vyšleme znak **b** nebo **B** je tento přijat druhým TCVR a odeslán do *arduina*. Na něm běžící kód jako reakci na příjem tohoto znaku provede na 500 ms bliknutí žlutou LED na desce *arduina* připojenou k pin 13.

Další možností je odeslání znaku **a** nebo **A.** Po jeho příjmu na straně *arduina* a jeho vyhodnocení kódem na *arduinu* dojde k odeslání řetězce **Arduino for CANSAT CZ** prostřednictvím TCVR připojeného k *arduinu* a jeho příjmu druhým TCVR a následným zobrazením terminálovým programem na PC. Tím je ověřena funkčnost obou TCVR. Poznamenávám, že oba TCVR byly nastaveny naprosto stejně, tj. nejenom kmitočet, modulace, rychlost přenosu na vf straně i na straně UARTu 9600 Bd, ale i stejné (dokonce původní, defaultní) hodnoty NET ID a NODE ID.

Jako terminálový program jsem odzkoušel jednak terminálový program, který je součástí vývojového prostředí *arduina*:



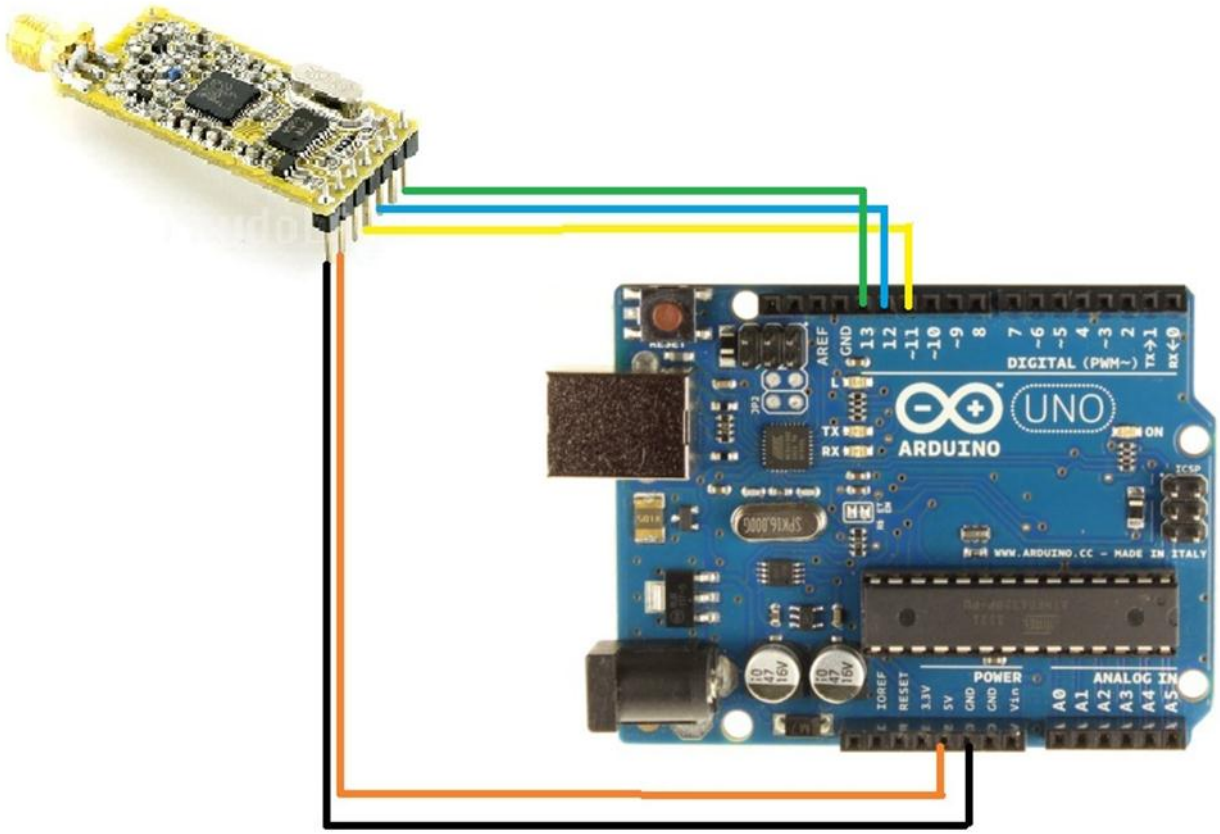
jednak RealnTerm:



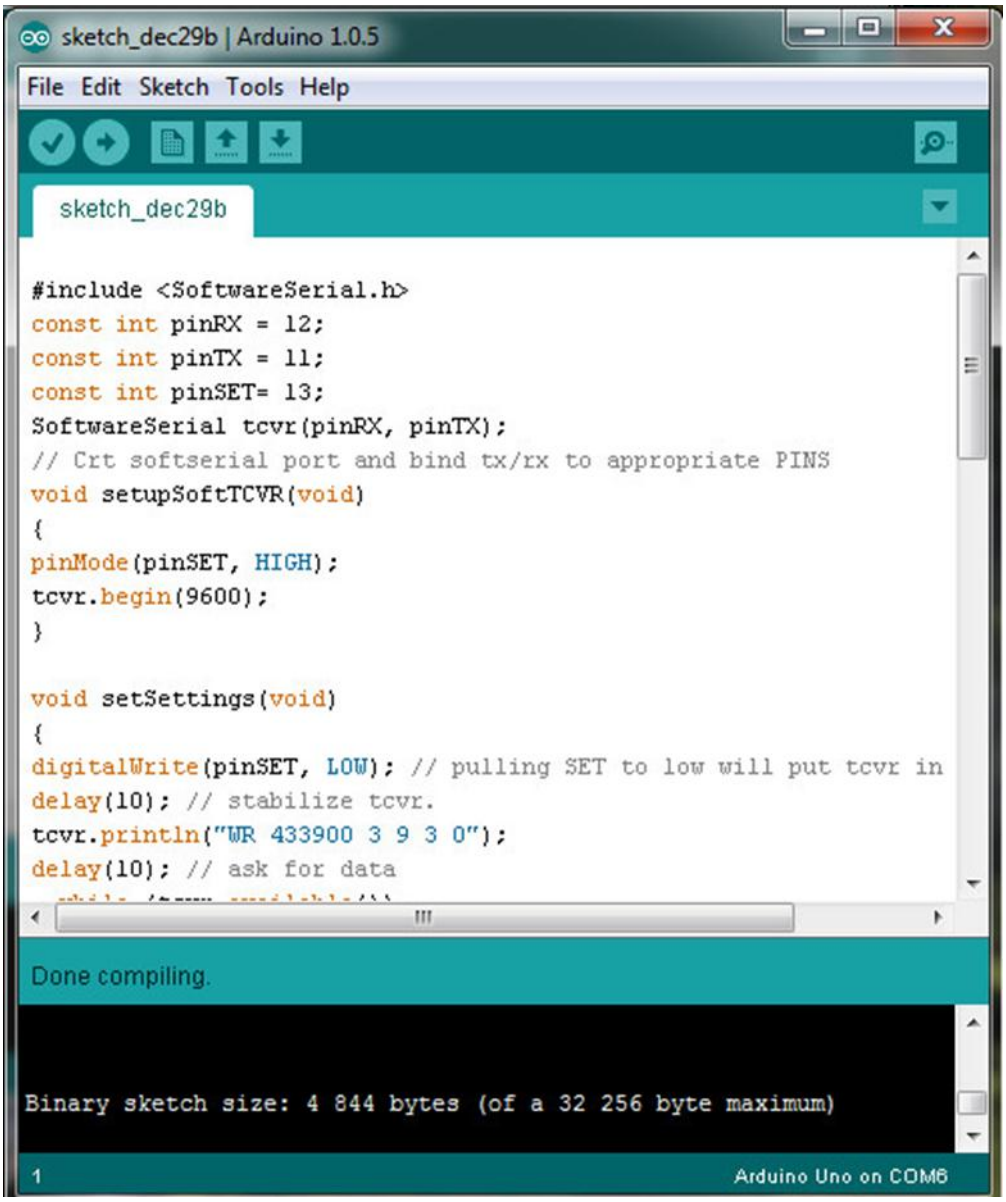
Vysílání obou TCVR jsem ještě sluchově sledoval příposlechem pomocí **TCVR Kenwood TM455** pro radioamatérské pásmo 70cm.

Nyní ukážu ještě jeden způsob konfigurace trancieverů APC220 využívajícího nastavení signálu SET (pin 7 tcvru) na L a následně přes RxD a TxD defaultní rychlostí 9600 Bd provádět příslušná nastavení. Využijeme k tomu *arduino*. Tentokrát ale připojíme TCVR k jeho pinům 11, 12 a 13:

pinRX = 12 pinTX = 11 a pinSET= 13



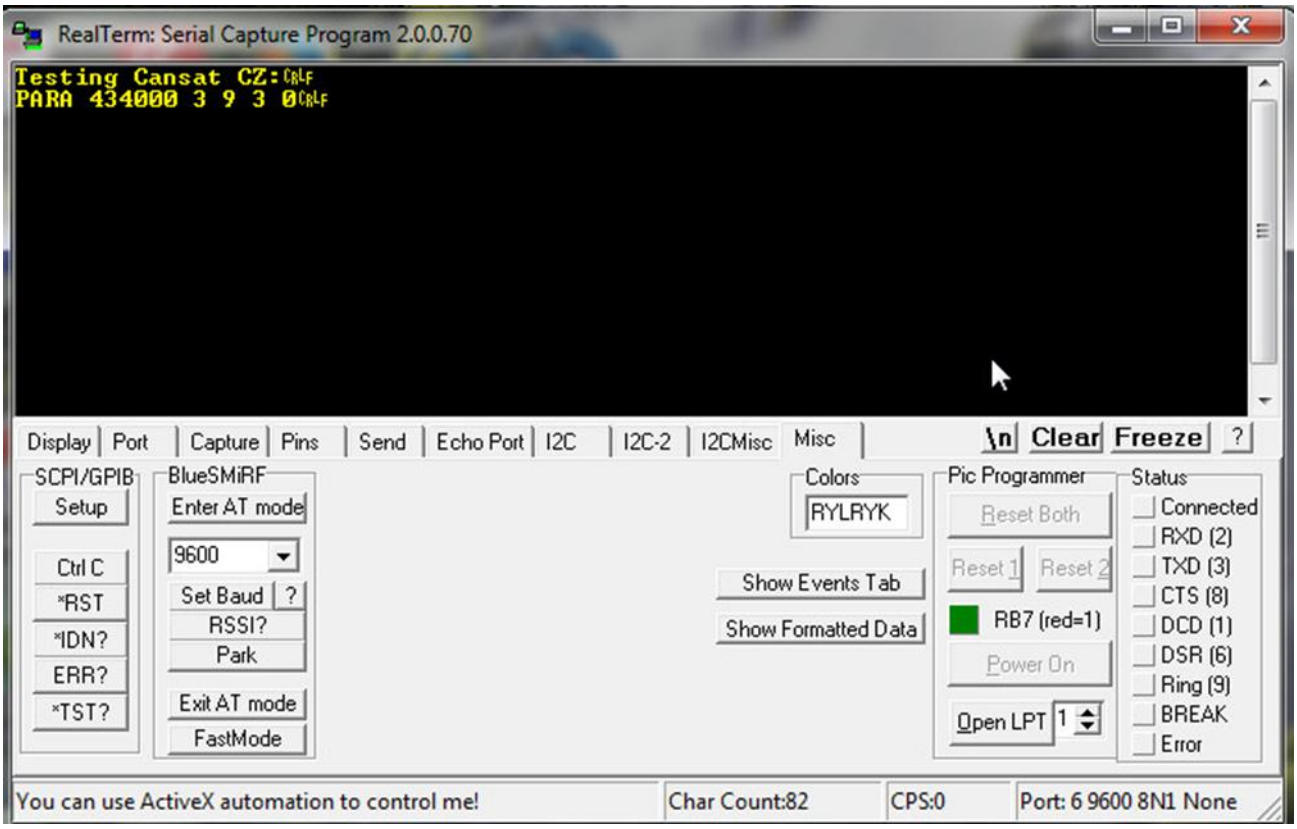
Do *arduina* nahrajeme kód:

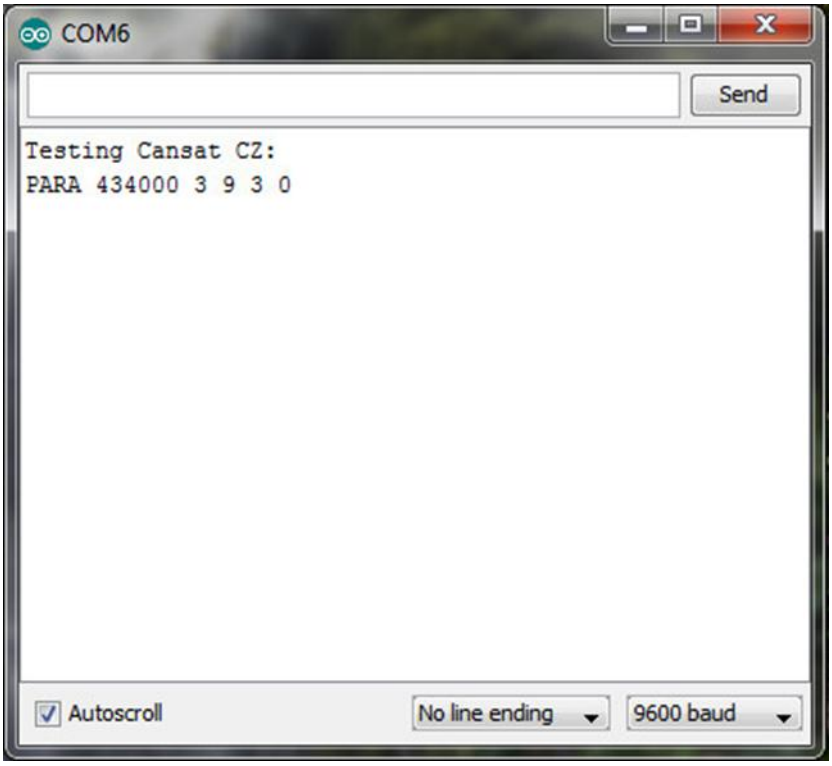



```
#include <SoftwareSerial.h>
const int pinRX = 12;
const int pinTX = 11;
const int pinSET= 13;
SoftwareSerial tcvr(pinRX, pinTX);
// Crt softserial port and bind tx/rx to appropriate PINS
void setupSoftTCVR(void)
{
pinMode(pinSET, HIGH);
tcvr.begin(9600);
}

void setSettings(void)
{
digitalWrite(pinSET, LOW); // pulling SET to low will put tcvr in config mode
delay(10); // stabilize tcvr.
tcvr.println("WR 433900 3 9 3 0");
delay(10); // ask for data
  while (tcvr.available())
```

```
    {
       Serial.write(tcvr.read());
    }

digitalWrite(pinSET, HIGH); // put tcvr back in operation
delay(200);
}

void getSettings(void)
{
digitalWrite(pinSET, LOW); // pulling SET to low will put tcvr in config mode
delay(10); // stabilize tcvr
tcvr.println("RD");
delay(10); // ask for data
while (tcvr.available())
  {
    Serial.write(tcvr.read());
  }
digitalWrite(pinSET, HIGH); // put tcvr back in operation
delay(200);
}

void setup()
{
Serial.begin(9600);
setupSoftTCVR();
Serial.println("Testing Cansat CZ:");
getSettings();
//setSettings();
}

void loop()
{
tcvr.println("Hello Cansat CZ!");
delay(5000);
}
```

Program pracuje tak, že jeho hlavní část proběhne hned na začátku, kdy se provádí funkce setup(). Ta nejprve nastaví přenosovou rychlost sériového kanálu (přes který komunikuje *arduino* s PC) na 9600 Bd. Na tuto rychlost nastaví rovněž softwarově vytvořený sériový kanál, k němuž je připojen TCVR. Nastavení přenosové rychlosti softwarového sériového kanále se provede voláním funkce `setupSoftTCVR`. Následně se volá funkce `getSettings`. Ta nejprve nastaví na L úroveň pinu SET (7) TCVR. Poté odešle přes softwarový sériový kanál do TCVR řetězec znaků **RD** následovaný řídícími znaky cr lf (hexadecimálně 0x0D 0x0A). Protože při SET=L je TCVR v konfiguračním režimu, není tento řetězec odvysílán přes vf, ale je chápán jako příkaz, aby odeslal zpět informace o svém nastavení.
To také tcvr provede (v našem případě odeslal řetězec znaků **PARA 434000 3 9 3 0** následovaný řídícími znaky cr lf). Přijatý řetězec je poté odeslán přes druhý sériový kanál do PC, kde je zobrazen na terminálovém programu. Odzkoušel jsem jak *RealTerm*, tak terminálový program obsažený ve vývojovém prostředí *arduina:*



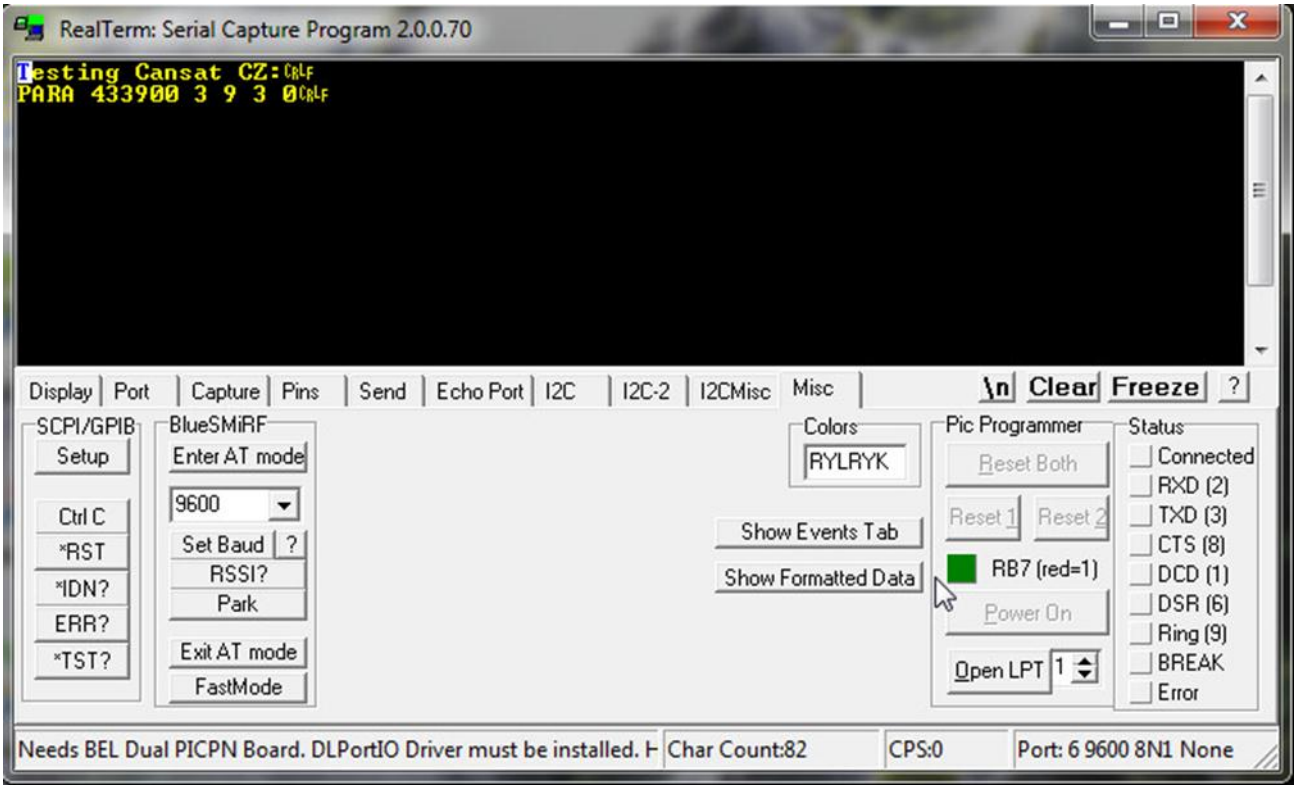


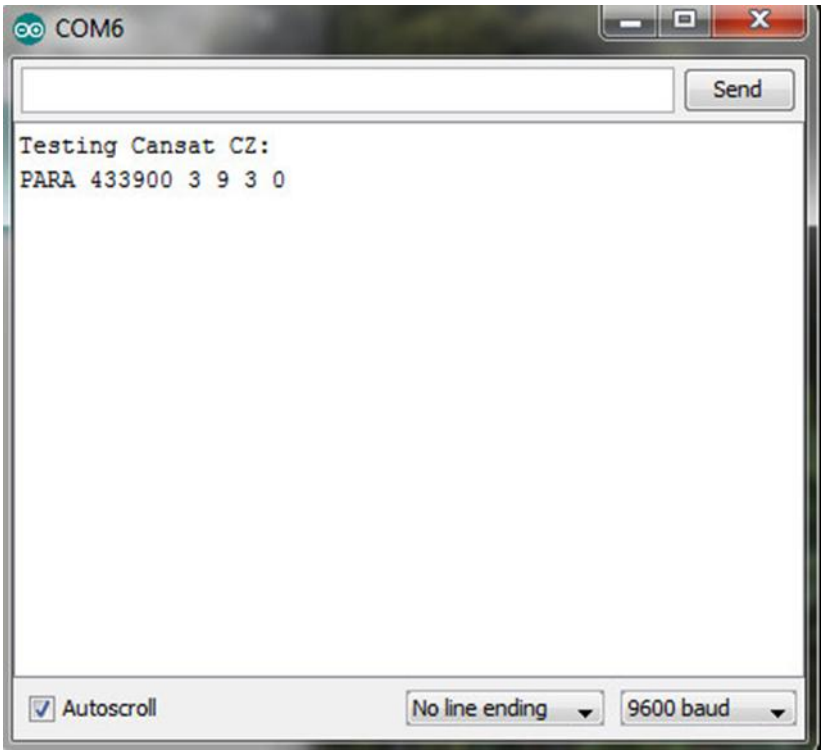
Místo příkazu **RD** pro tcvr v konfiguračním režimu můžeme pro něj použít příkaz **WR** s příslušnými parametry (v našem případě **WR 433900 3 9 3 0**) a tak nastavit tcvr. Není tedy nutné používat k nastavení tcvr program **RF-Magic (for APC22x v.1.2A)** či **DRF Tools for ADF702x series**.

Do *arduina* k tomuto účelu nahrajeme kód od výše uvedeného se lišícího jen ve funkci setup():

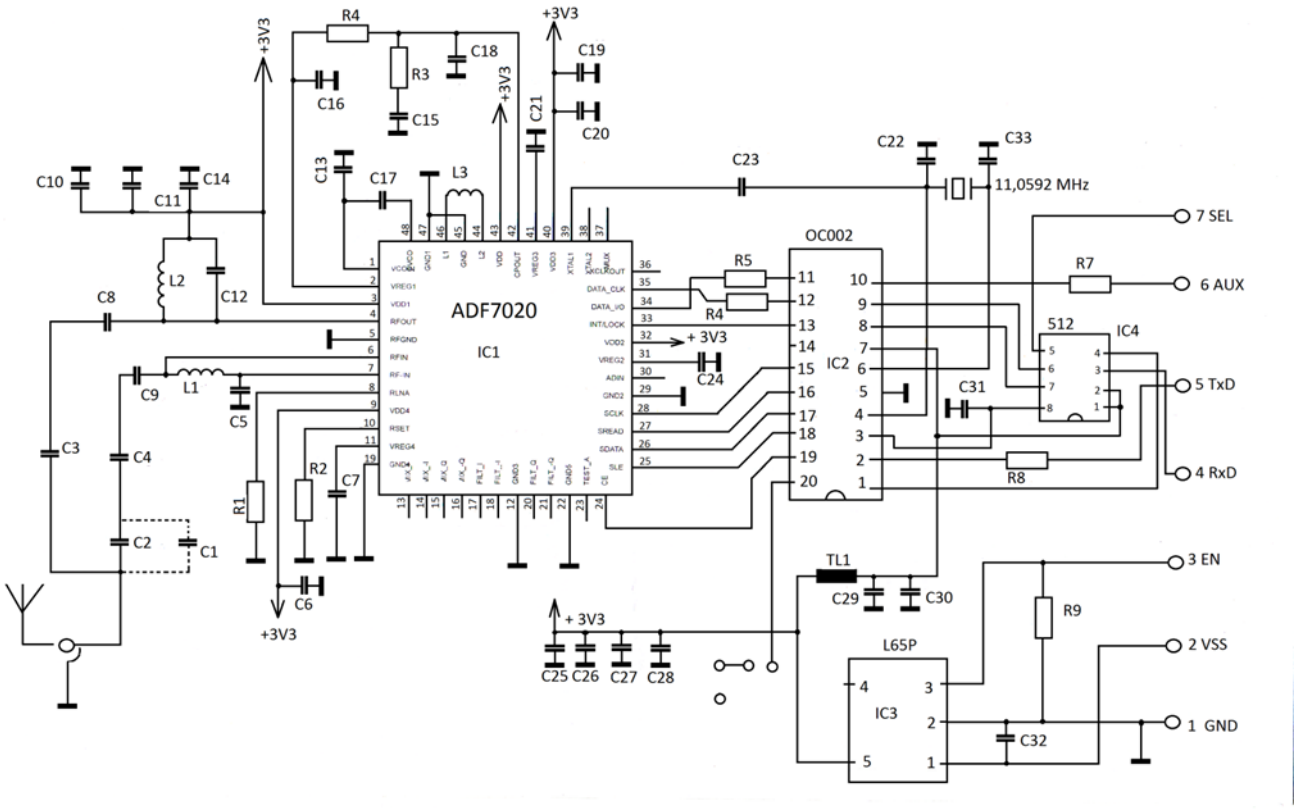
```
void setup()
{
Seriál.begin(9600);
setupSoftTCVR();
Seriál.println(„Testing Cansat CZ:“);
//getSettings();
setSettings();

}
```

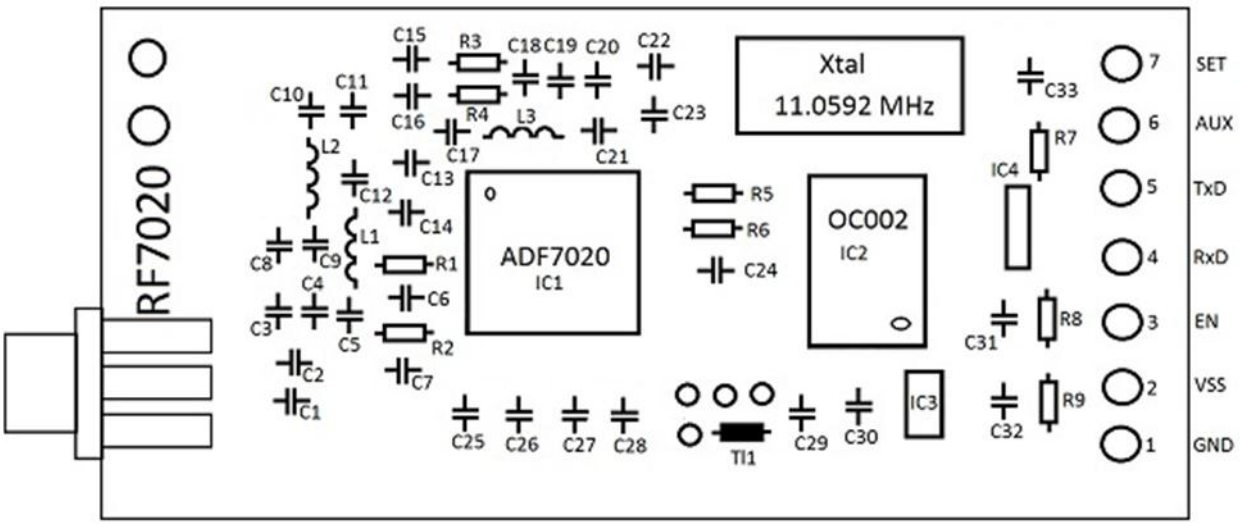
Na terminále se objeví





neboť se po změně konfigurace tcvr ještě provede její přečtení a následné odeslání na terminál. Ještě si vysvětlíme význam parametrů u WR a PARA. První parametr v **PARA 433900 3 9 3 0** je zřejmý. Je to kmitočet tcvr 433.9MHz. Druhý parametr je celé číslo 1, 2, 3 či 4 udávající přenosovou rychlost 2400, 4800, 9600 či 19200 Bd (b/s popř. bps) na vf (FSK). V našem případě máme nastaveno 9600 b/s. Třetí parametr je celé číslo 0, 1 až 9 udávající výstupní výkon. Nule odpovídá -1dBm, devítce pak maximální výkon 13dBm (20mW). My tedy máme nastavený maximální výstupní výkon. Čtvrtý parametr je celé číslo 0, 1, 2, 3, 4, 5 nebo 6 odpovídající rychlostem 1.2 , 2.4 , 4.8 , 9.6 , 19.2 , 38.4 nebo 57.6 kb/s kterou se s tcvr komunikuje přes uart (piny RxD a TxD). I zde máme nastavenu rychlost 9600 Bd. Poslední parametr 0, 1 nebo 2 udává paritu. Nula znamená bez paritního bitu.
Pokud umíme pracovat s arduino, měla by nám tato kapitola 2 stačit k úspěšné práci na projektu CanSAT pokud jde o programovou obsluhu bezdrátového spojení v pásmu 433MHz mezi CanSATem a pozemní stanicí.
Vzhledem k tomu, že jsem neměl k dispozici podrobnější údaje o tcvr, nakreslil jsem na základě našeho exempláře trancieveru jeho schéma a rozložení součástí:





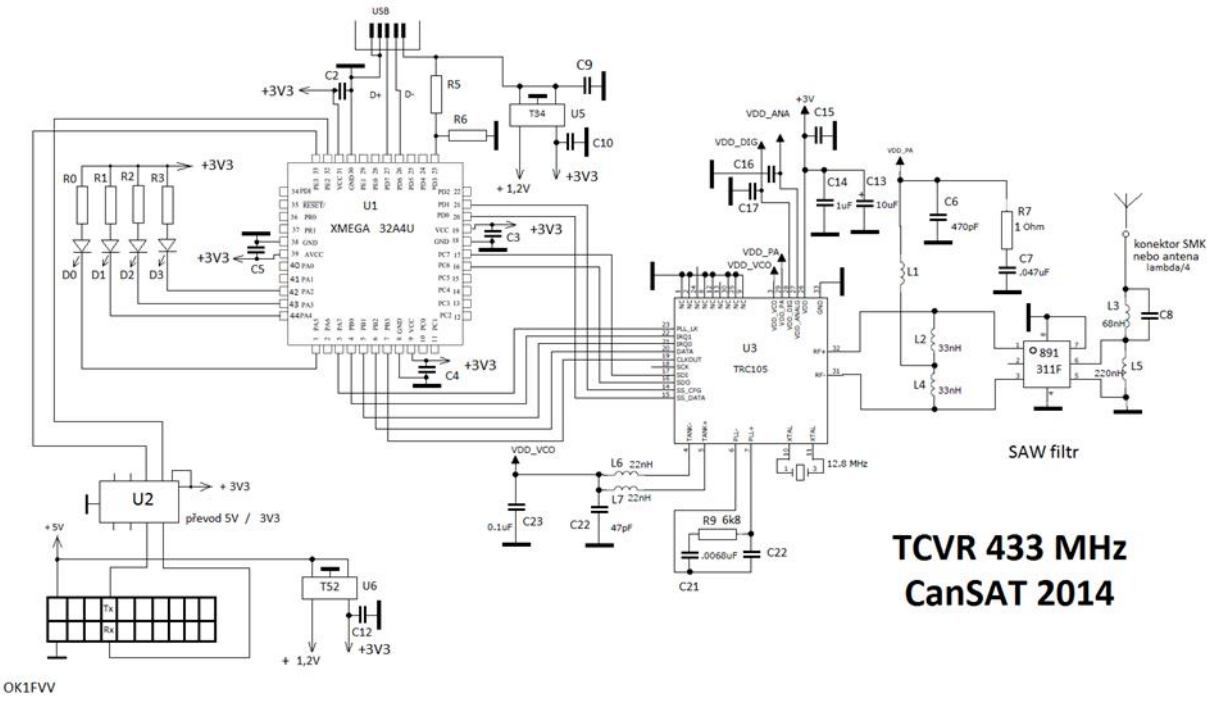
Tranciever se tedy skládá ze dvou částí – jednočipového trancieveru ADF7020 a mikrořadiče. Obvod ADF7020 je zapojen v doporučeném zapojení výrobce *Analog Devices.*

4.2 Tcvr T-minus Engineering

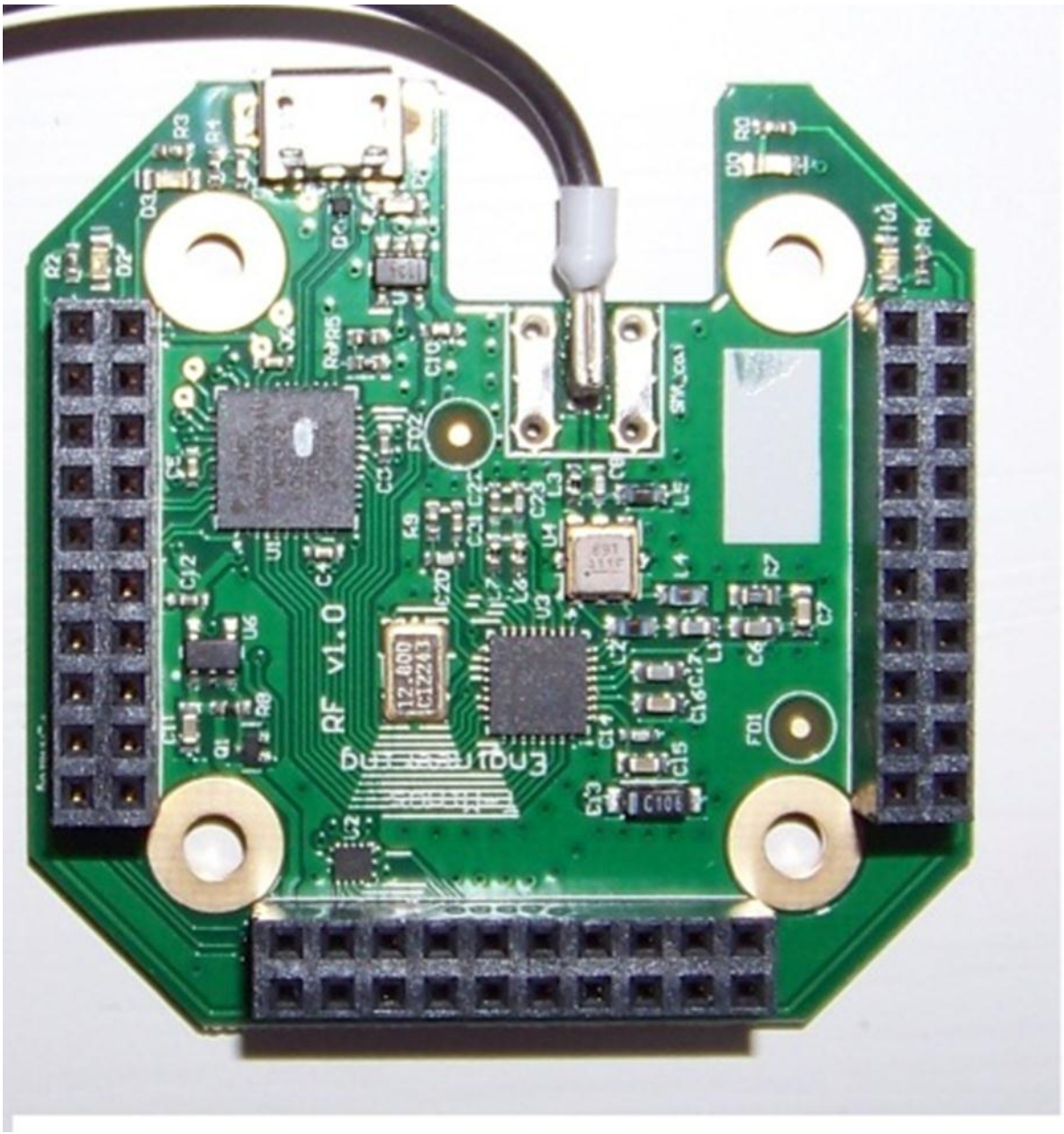
Základem trancieveru je obvod TRC105 firmy RFM. Vysílač může pracovat se dvěma druhy modulace buď OOK (on-off keyed) nebo FSK (frequency-shift keyed). Je-li nastaveno OOK

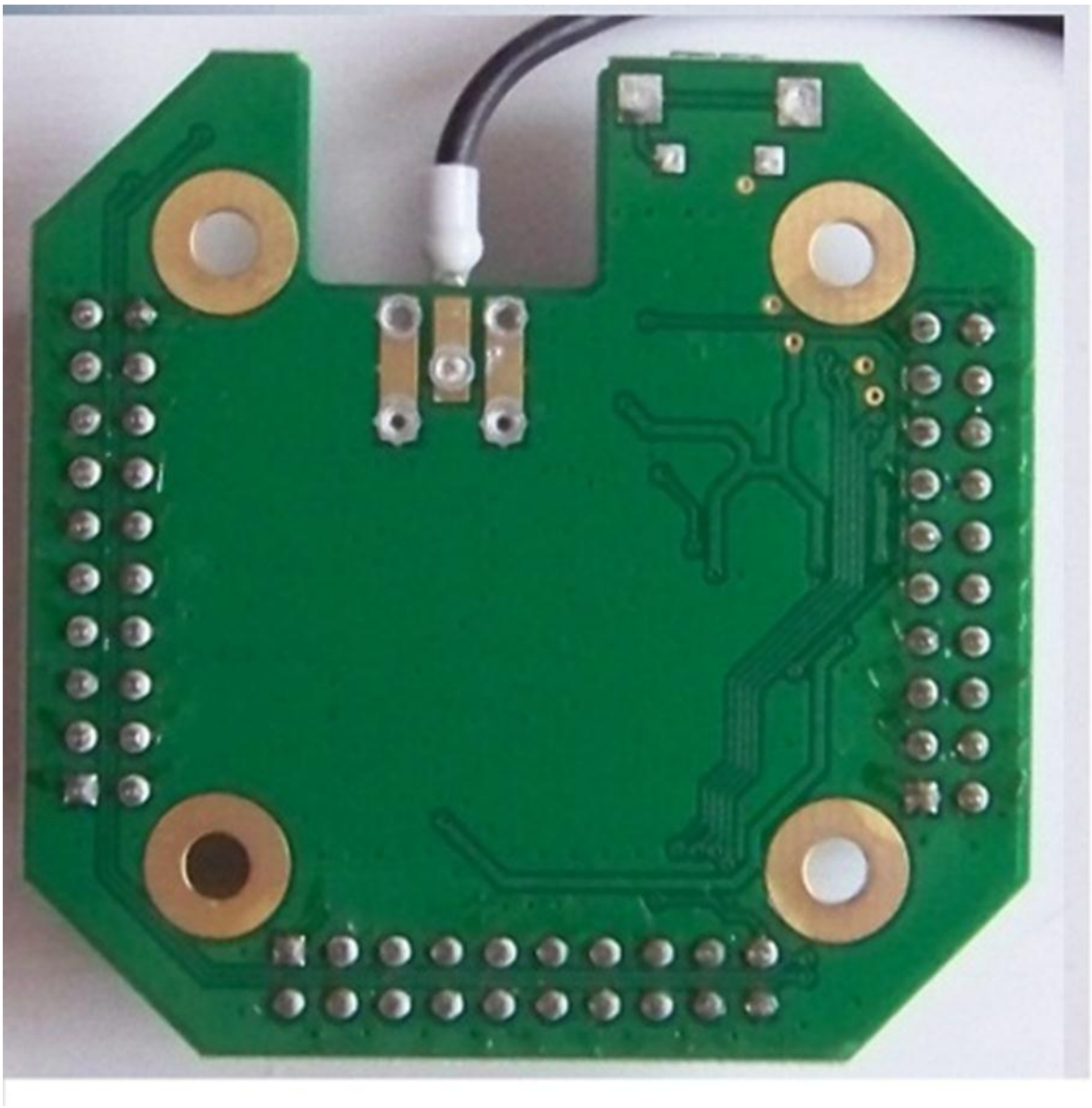
pak log 1 znamená plný výkon a log 0 je bez výkonu. Je to tedy obdoba provozu A1. Napájecí napětí je 2.1 až 3.6V. Citlivost přijímače -112 dBm. Maximální výkon vysilače je +13 dBm. Řídícím mikrokontrolerem je u tohoto tcvr XMEGA32A4U. Tranciever pracuje na jednom z 24 kanálů v pásmu 70 cm s roztečí 90kHz mezi kanály. Jejich kmitočty jsou:

432.99 MHz	433.53 MHz	434.07 MHz	434.61 MHz
432.08 MHz	433.62 MHz	434.16 MHz	434.70 MHz
433.17 MHz	433.71 MHz	434.25 MHz	434.79 MHz
433.26 MHz	433.80 MHz	434.34 MHz	434.89 MHz
433.35 MHz	433.89 MHz	434.43 MHz	434.97 MHz
433.44 MHz	433.98 MHz	434.52 MHz	435.06 MHz

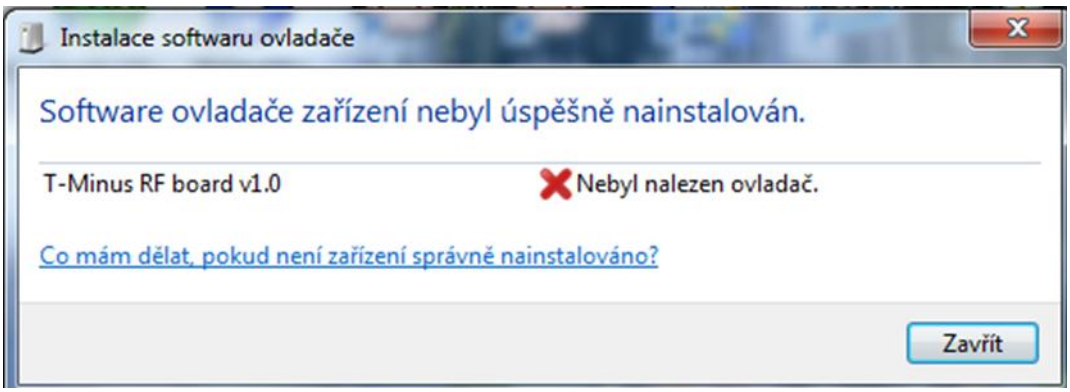


Oba dva obvody U1 a U3 jsou napájeny 3V3. Toto napětí získáme při napájení přes usb konektor (+5V) ze stabilizátoru U5, při napájení 5V z konektoru od bloku s MCU ATMega2560 opět ze stabilizátoru, tentokrát U6. Dalšími signály z konektoru jsou Tx a Rx (přes konektor spojené s RxD a Txd ATmega 2560) na úrovni TTL (5V) z Arduina. Obvod U2 slouží pro jejich převod na úroveň 3V3 XMega 32A4U.

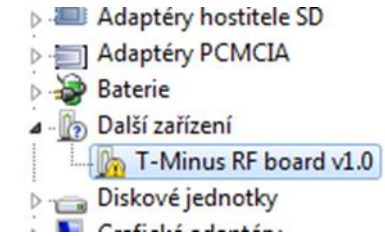

RF v1.0



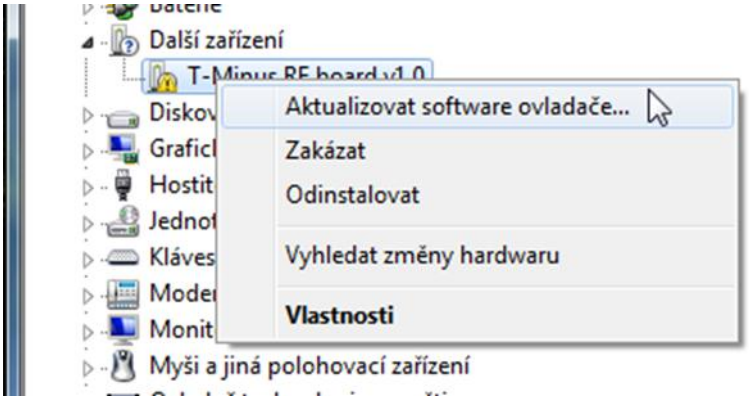
Pokud poprvé připojíme tcvr přes usb k PC (např. s WIN7), OS zjistí nové usb zařízení a pokusí se (neúspěšně) nainstalovat drivery:



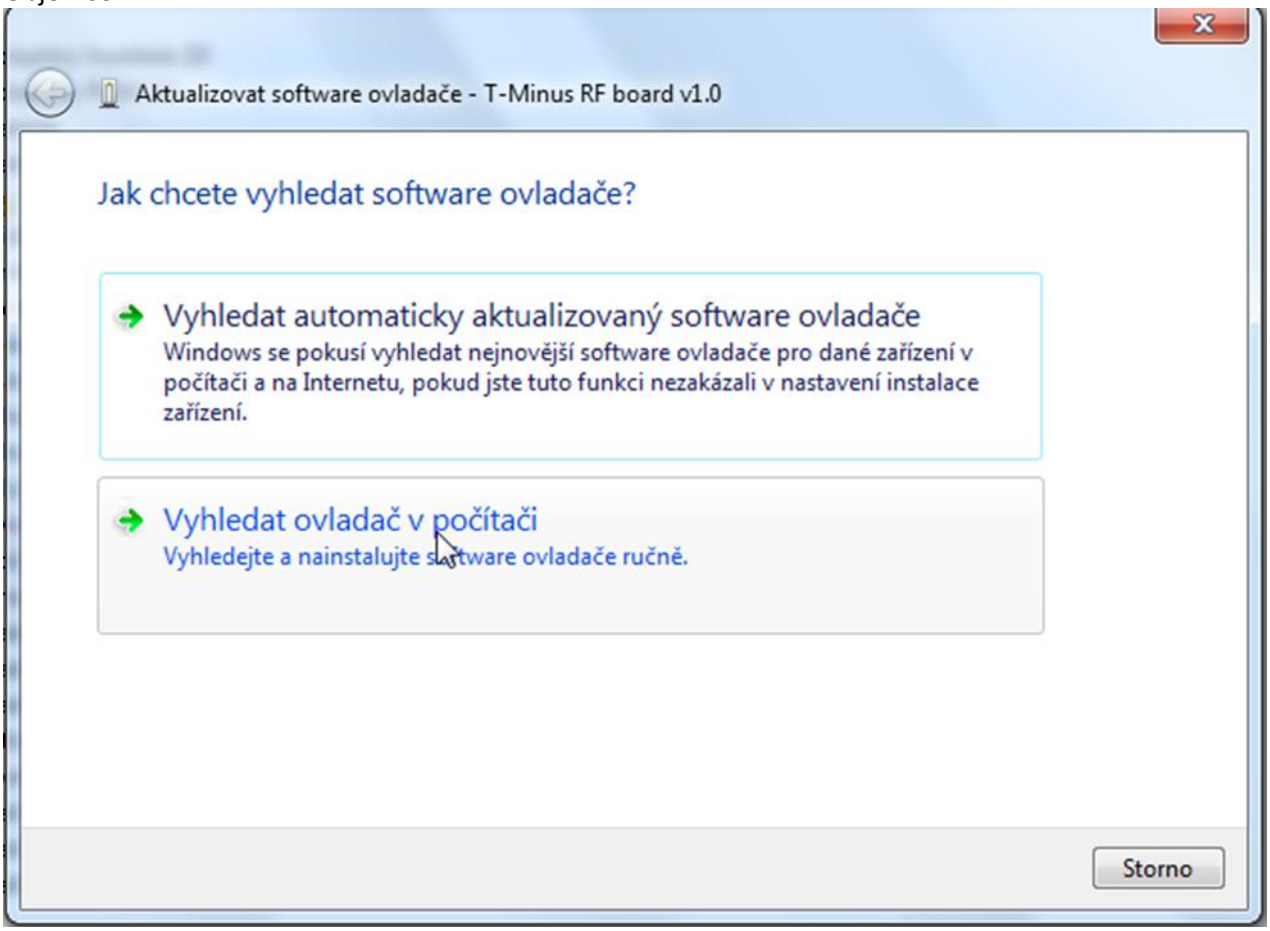
Ve správci zařízení uvidíme:



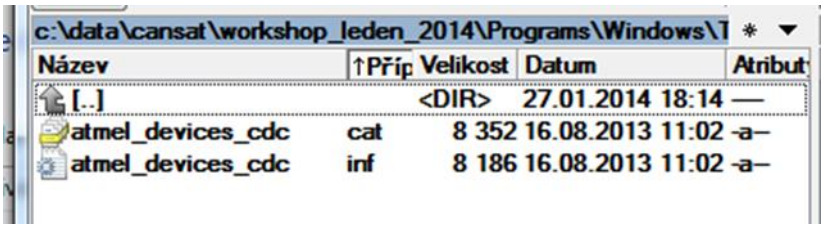
Pravým tlačítkem rozvineme lokální menu nad položkou T-Minus RF board v1.0:



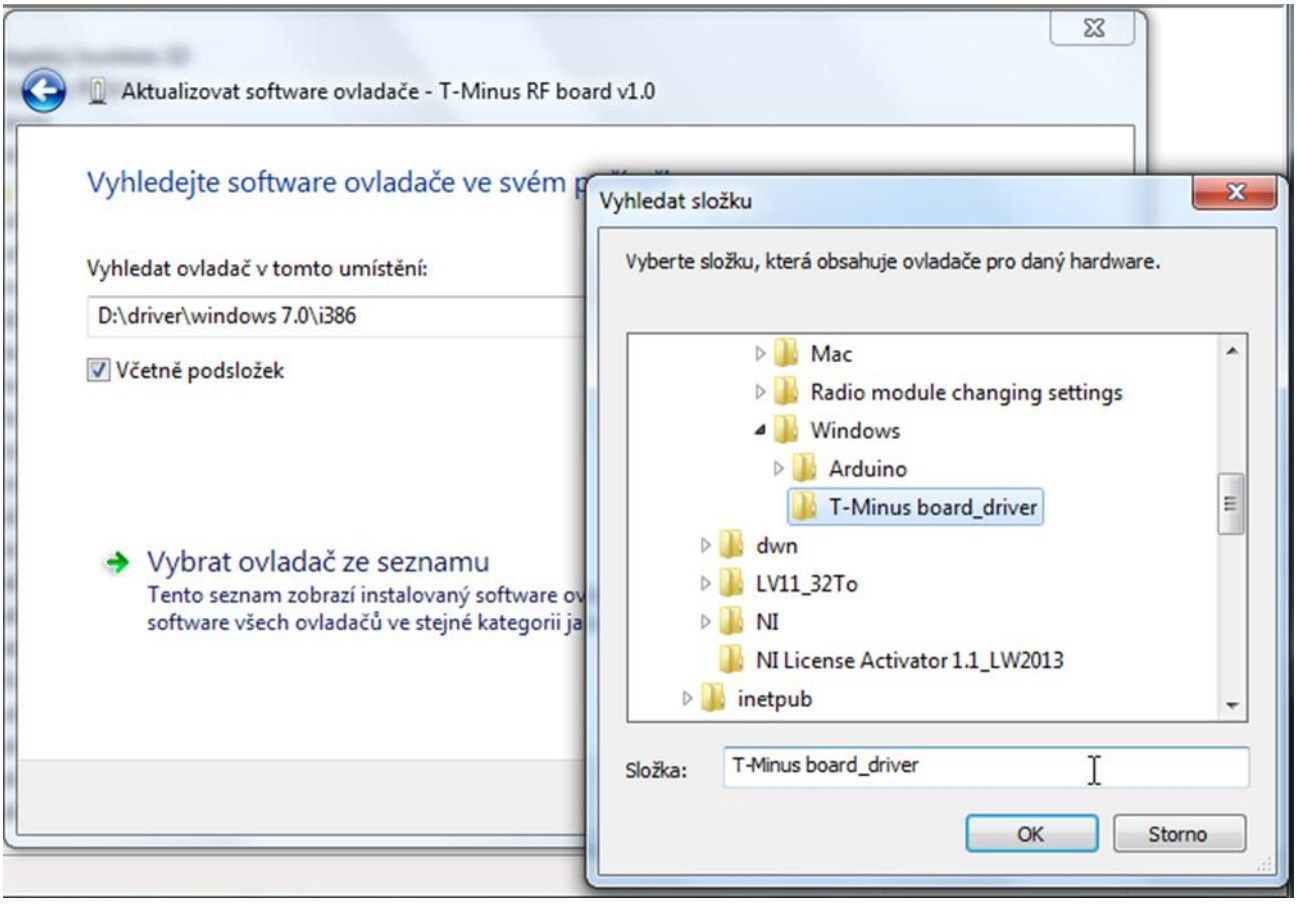
Objeví se



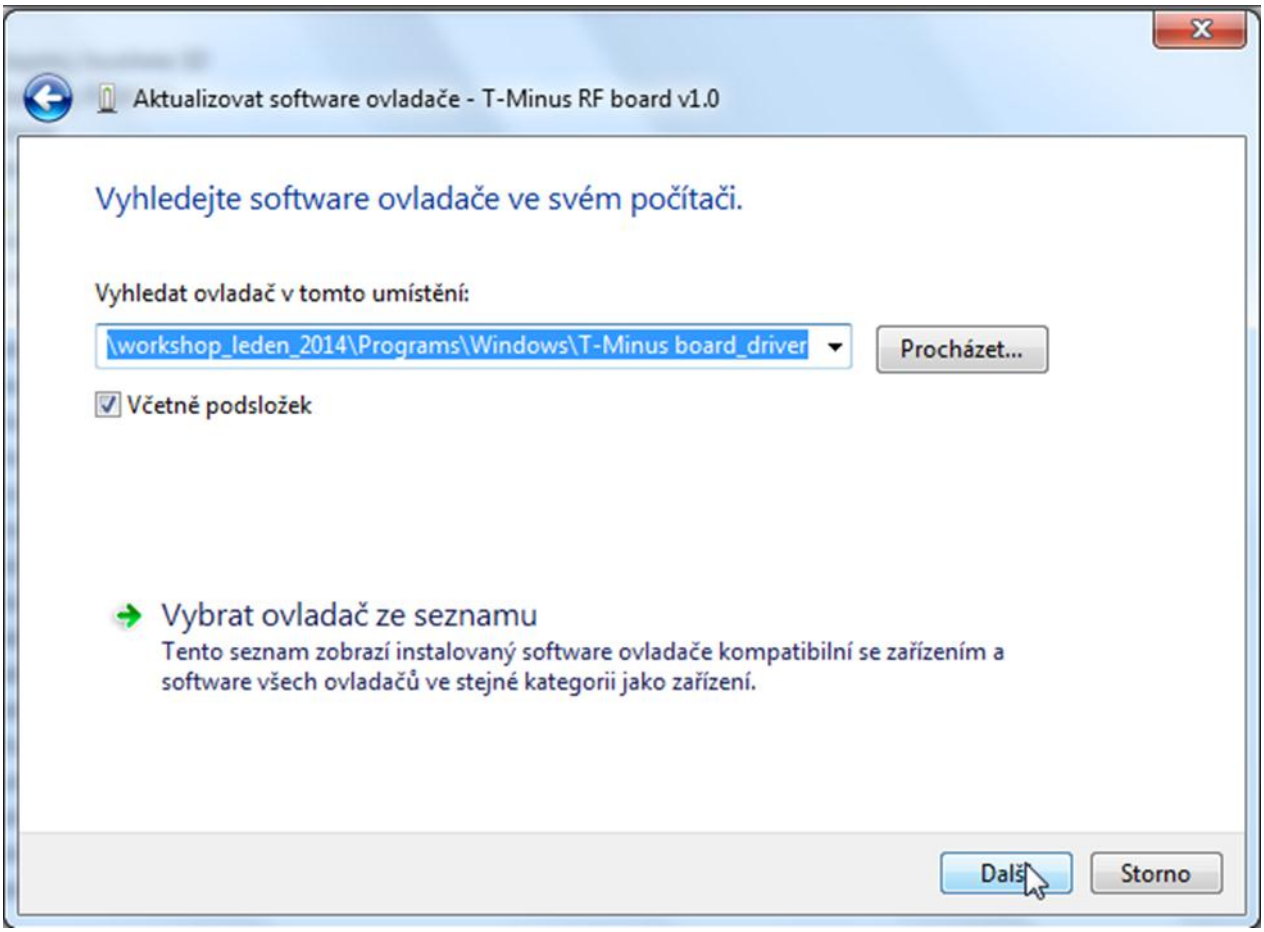
Vybereme položku Vyhledat ovladač v počítači. Tento ovladač jsme získali na workshopu v ESA ESTAC a jmenuje se atmel_devices_cdc (stejný ovladač budeme používat i pro arduino TMinus1 na desce palubního počítače s ATMega 2560)



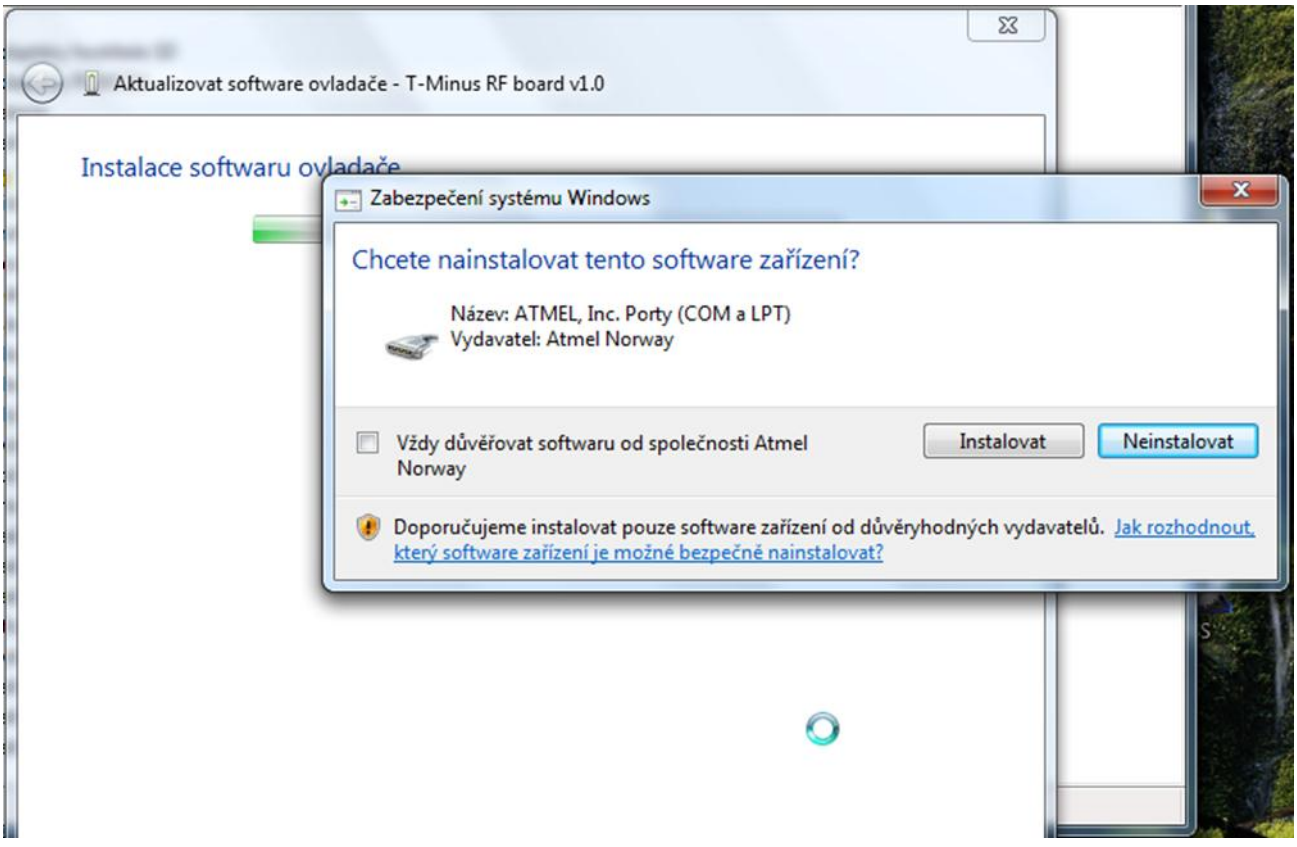
Nastavíme cestu na tento ovladač:



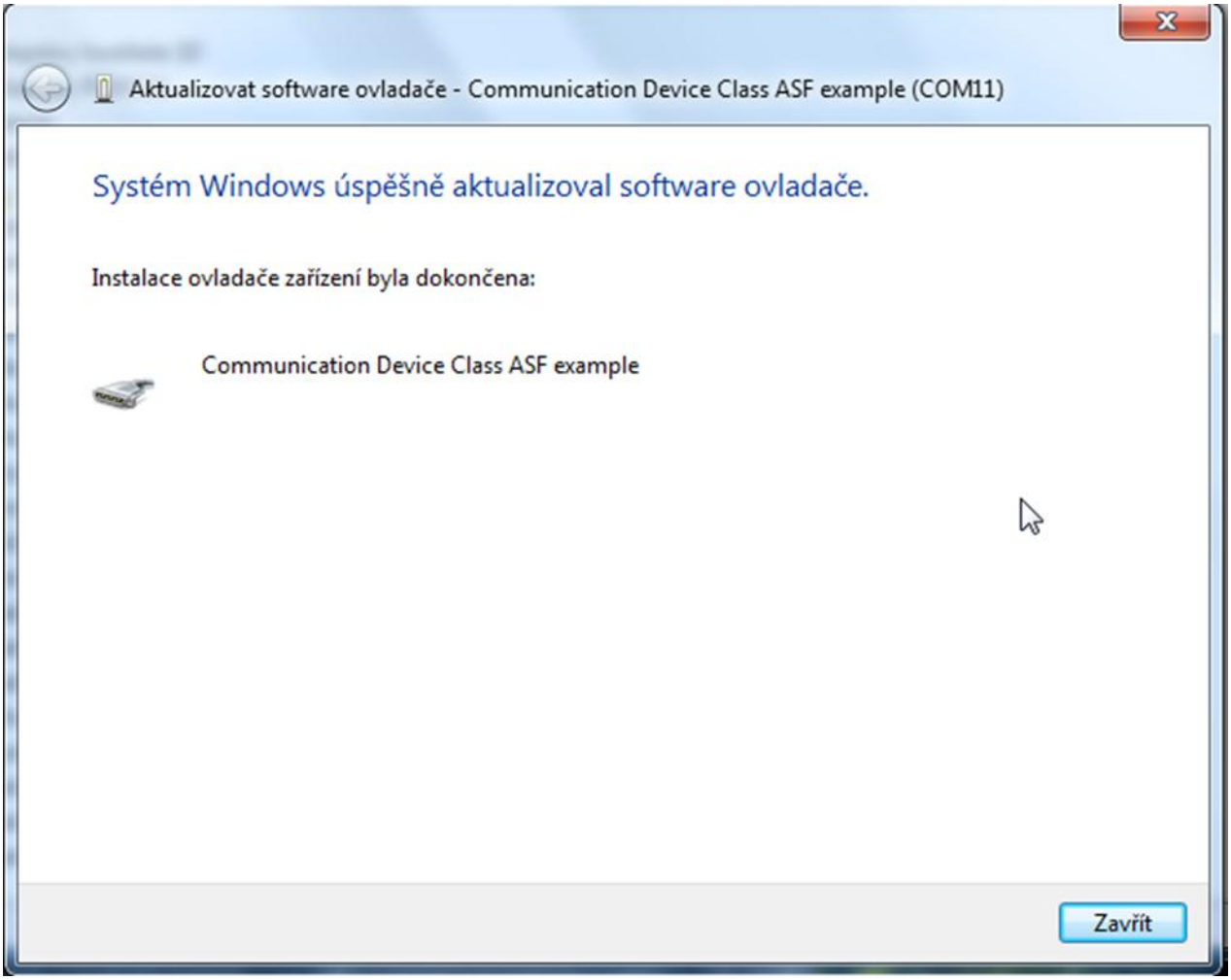
Potvrdíme **OK.**



Klikneme na **Další**. Objeví se:

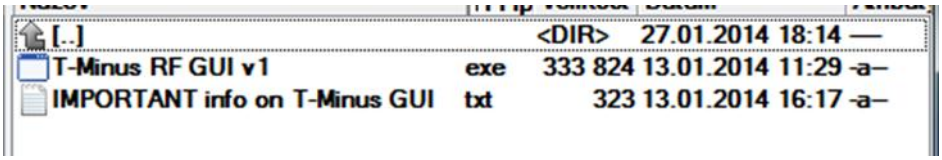


Klikneme na **Instalovat**.

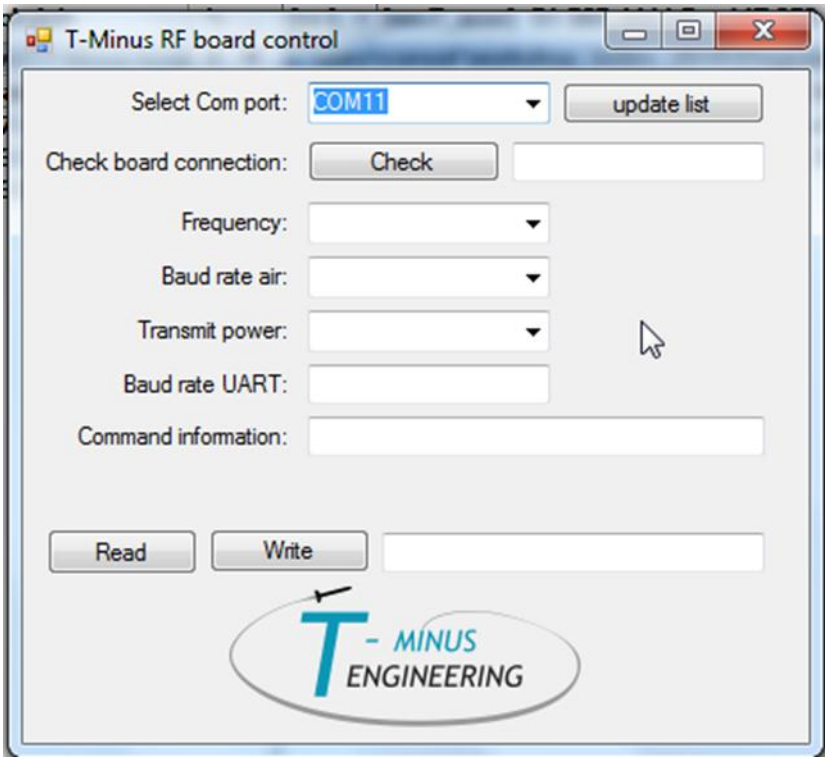


Driver je úspěšně nainstalován. Ve správci zařízení se objeví virtuální port se zařízením **Communication Device Class ASF**

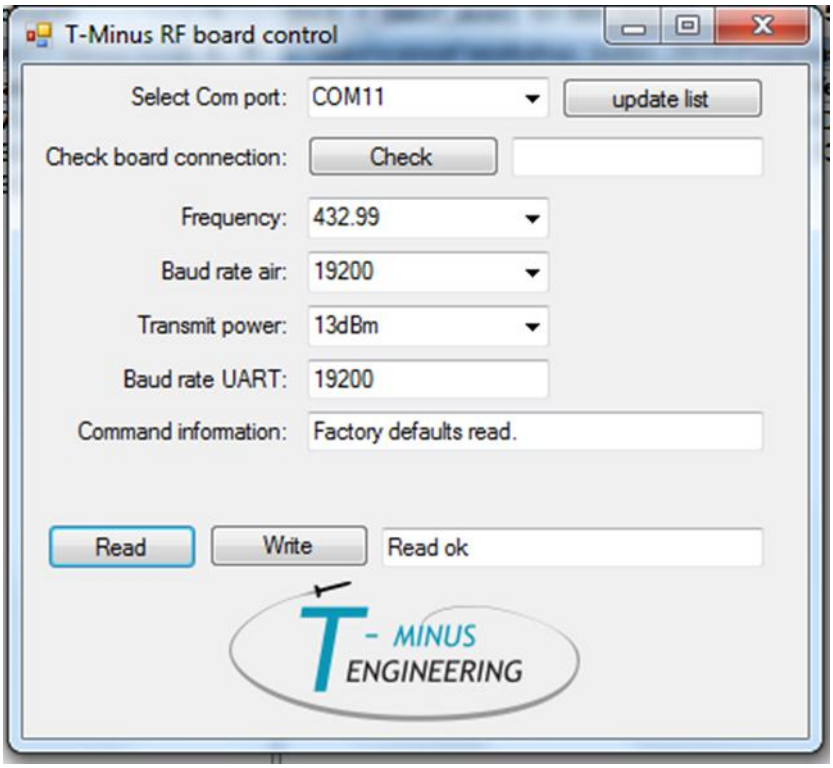
Nyní již může tcvr přes usb komunikovat s našim PC. Tuto komunikaci použijeme jednak v případě, že tcvr budeme používat jako pozemní stanici, jednak v případě nastavování tcvr pomocí programu T-Minus RF GUI v1. Tento program potřebuje na PC mít nainstalovaný .NET Framework, což v mém případě již bylo splněno, neboť na PC mám nainstalováno MS Visual Studio 2010.



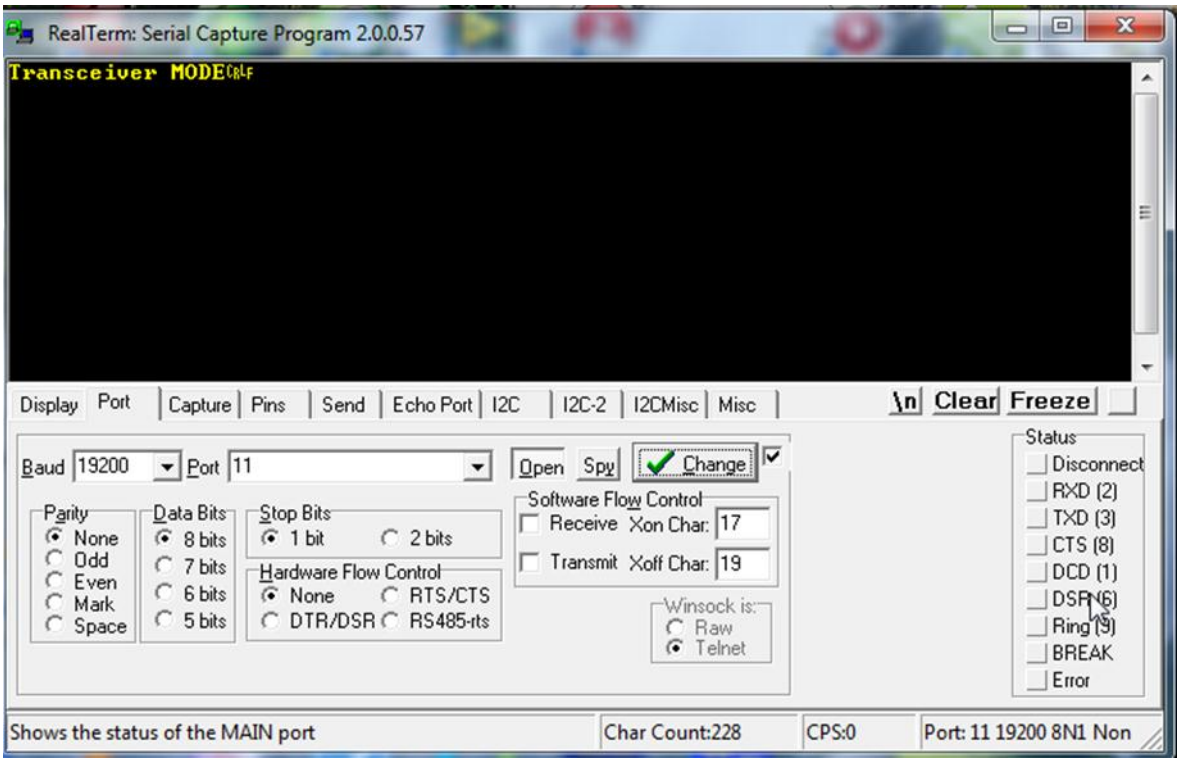
Spustíme-li zmíněný program, dostaneme:



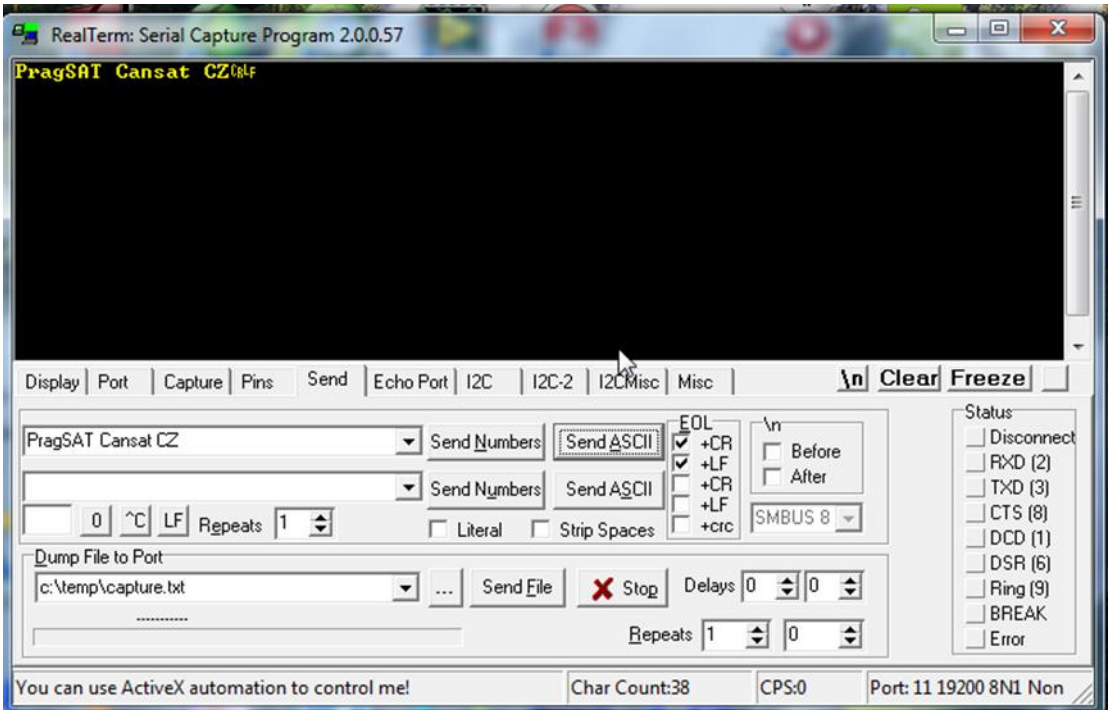
V horní nabídce **Select Com port** vybereme virtuální port odpovídající tcvr. Poté klikneme na tlačítko **Read**, dostaneme:



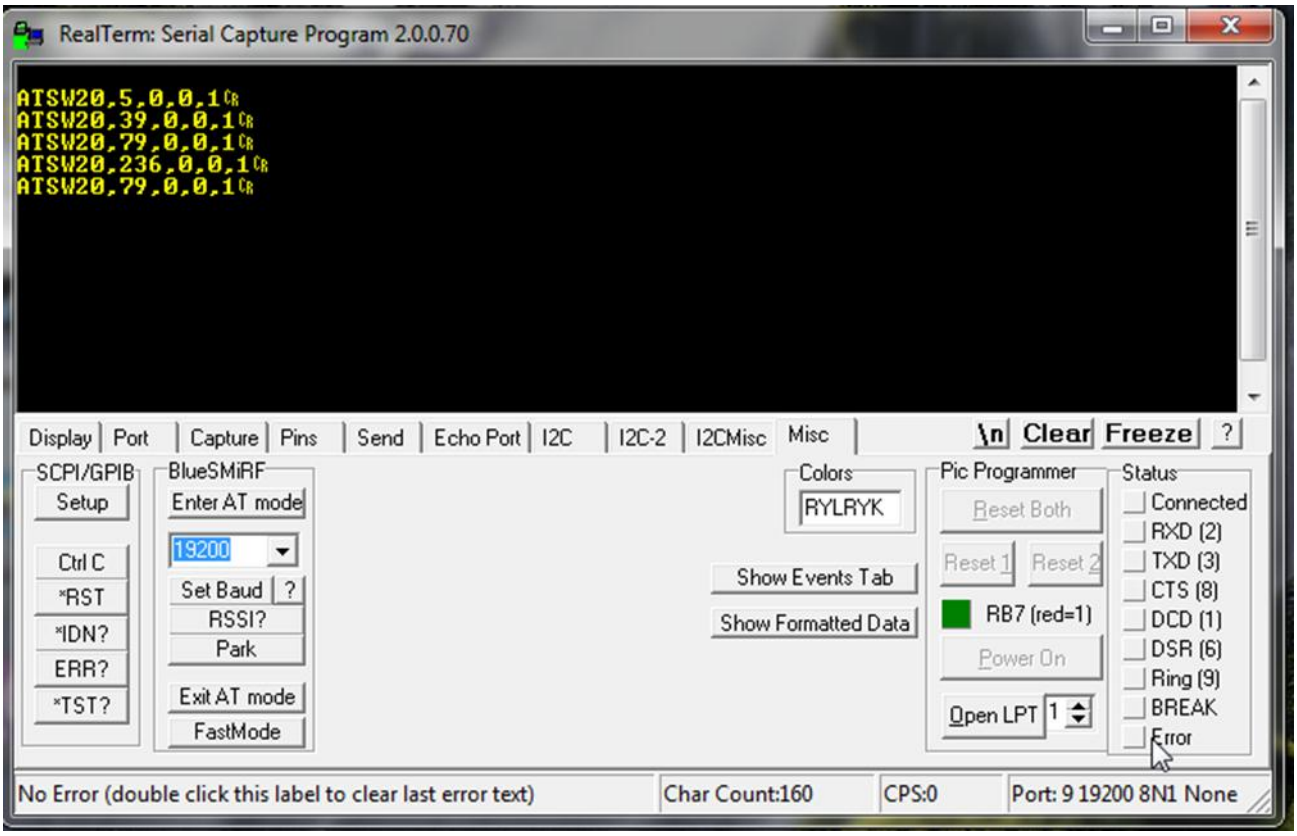
Jde o defaultní hodnoty. Můžeme je změnit a zapsat kliknutím na tlačítko **Write**. Dále můžeme odzkoušet komunikaci obou tcvr. Použil jsem dva notebooky (s nainstalovanými drivery) a k nim přes usb připojené tcvr. Na počítačích jsem spustil terminálový program s nastavenou rychlostí 19200 a odpovídajícím virtuální portem. Tranciever se nejprve ohlásí řetězcem Tranciever MODE:



Poté již lze mezi oběma tcvr posílat řetězce znaků:



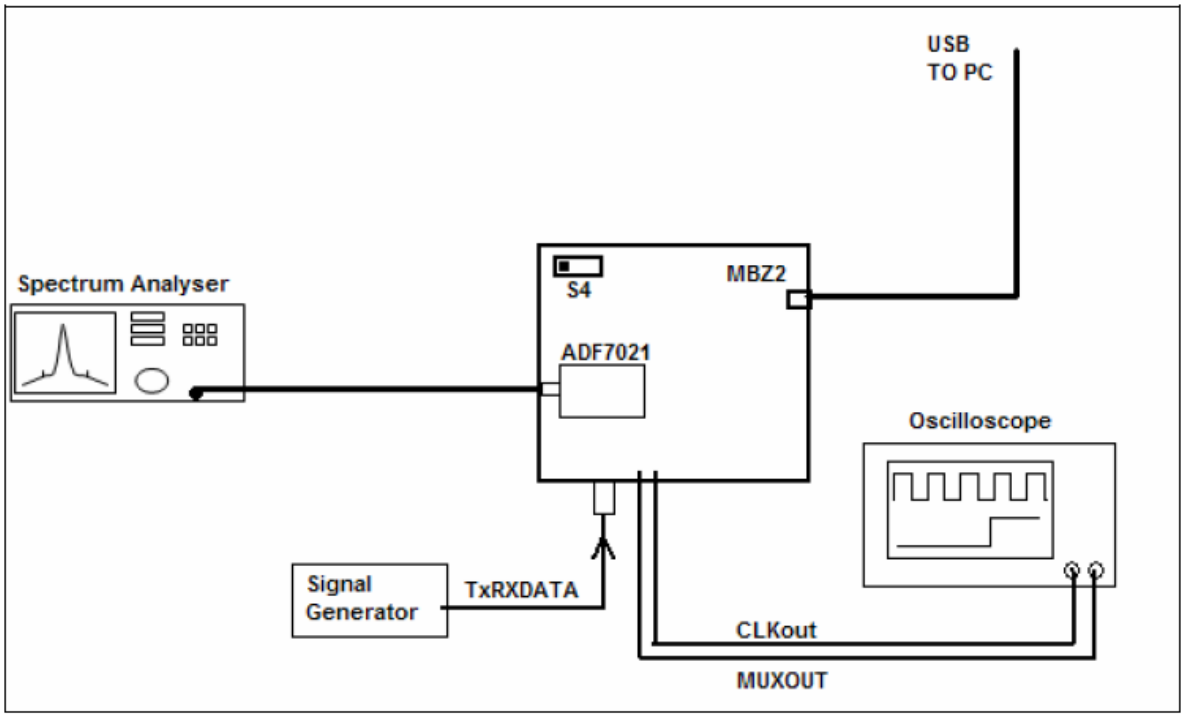
Pozn.:
Při změně nastavení rychlosti na terminálovém programu reaguje tcvr odesláním řetězce ATSW20 a nějakými dalšími čísly, přičemž první z nich závisí na nastavení rychlosti na terminálovém programu. Další číslice byly 0 0 a 1. Takto jsem postupně nastavil čtyři různé rychlosti terminálového programu a dostal tak:



4.3 Vlastní konstrukce tcvr s ADF7021

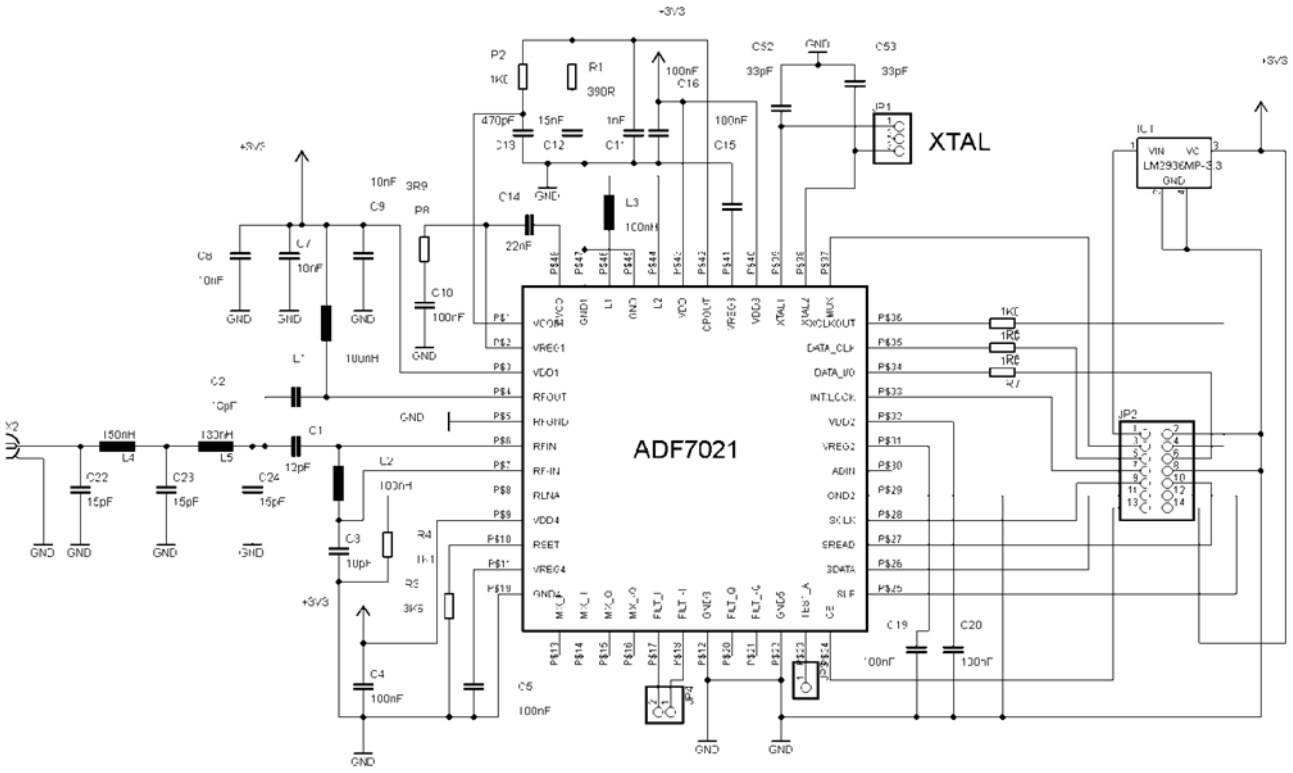
Při konstrukci tohoto tcvr jsem vycházel z doporučeného zapojení výrobce Analog Devices uvedeného v aplikačních listech. Pro vlastní konstrukci, rozložení součástí mi byla inspirací destička EVAL-ADF7021-NDBZ5:

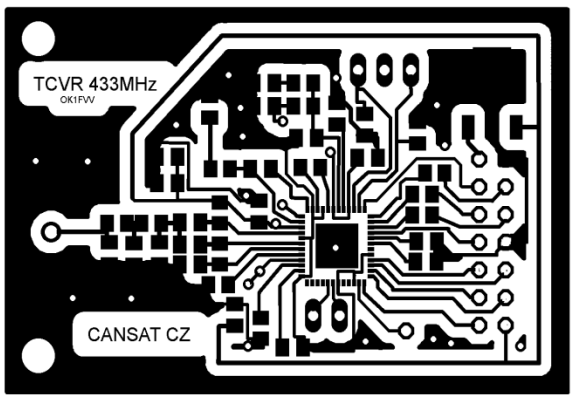
Je součástí měřícího pracoviště:

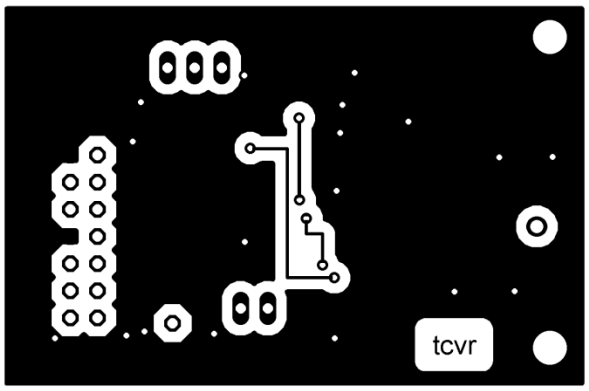


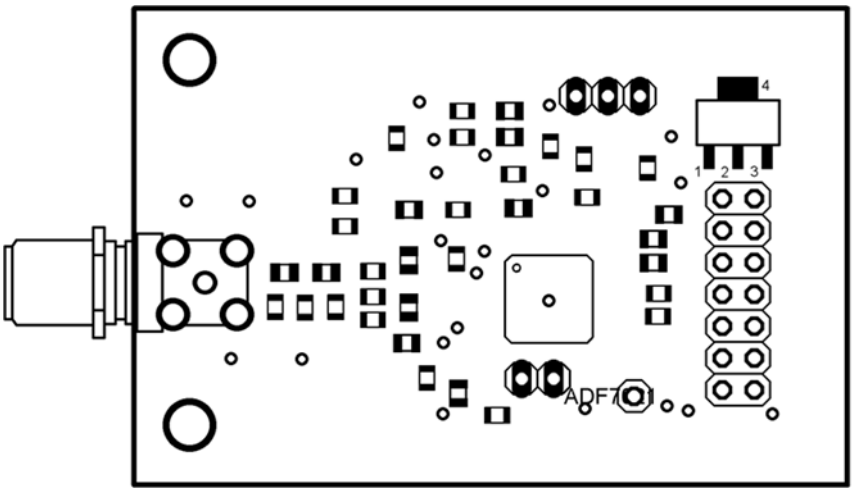
Deska MBZ2 je nazývána Evaluation Board. Menší destičky (viz předchozí foto) do ní pomocí konektoru připojované se liší pracovní frekvencí, popř. i IO.

Výsledkem mého snažení je zapojení a PCB :









Zapojení konektoru:

1	+ 5 V	2	GND
3	MUX	4	XCLKOUT
5	DATA_CLK	6	DATA_I/O
7	INT/LOCK	8	GND
9	SCLK	10	SREAD
11	SDATA	12	SLE
13	CE	14	+ 3V3

Hodnoty součástí společné pro všechna pásma (jsou převzata z aplikačních listů):

BILL OF MATERIALS

Table 3. Bill Of Materials for the EVAL-ADF7021DBX Daughter Boards (Common to all versions)

Qty	Name	Value	Tolerance	PCB Decal	Manufacturing Part No.
6	C4, C7, C9, C16, C20, C25	10nF	±10%	0402	
1	C21	22uF	±10%	0805	
1	C6	10uF	±10%	0805	
1	C14	22nF	±10%	0402	
1	C8	220pF	±5%	0402	
4	C5, C10, C15, C19	100nF	±10%	0402	
1	C17	22pF	±5%	0402	
3	C18, C22, C24	Not Inserted		0402	
1	J1			HEADER14	
1	J2			SMA_EDGE_RF	
2	R5, R6, R7	1k	±5%	0402	
1	R3	3.6k	±5%	0402	
1	R4	1.1k	±5%	0402	
1	R8	3.9r	±5%	0402	
1	U1			LFCSP-48	ADF7021BCPZ

Pozn.místo SMD 0402 lze použít 0805

Hodnoty součástí závislých na kmitočtu tcvr

Pásmo 433MHz (hodnoty z datasheetu Analog Devices):

Table 5. Bill Of Materials for components specific to EVAL-ADF7021DBZ3 Daughter Board

Qty	Name	Value	Tolerance	PCB Decal	Manufacturing Part No.
Matching					
1	C1	4.7pF	±0.25pF	0402	GRM1555C1H4R7CZ01D
1	C2	10pF	±5%	0402	GRM1555C1H100JZ01D
1	C3	6.8pF	±0.5pF	0402	GRM1555C1H6R8DZ01D
1	L1	13nH	±5%	0402	Coilcraft 0402CS-13NX-JLU
1	L2	27nH	±5%	0402	Coilcraft 0402CS-27NX-JLU
VCO External Inductor					
1	L3	Not inserted		0402	
Harmonic Filter					
1	L4	22nH	±5%	0402	
1	L5	20nH	±5%	0402	
1	C23	6.8pF	±0.5pF	0402	
	C22	Not inserted			
	C24	Not inserted			
TCXO					
1	Y1	19.68MHz	2.5ppm	5.0x3.2x1.3mm SMD	SIWARD TXO812025LJ-19.68MHz-3.0R
Loop Filter					
1	R1	270r	±10%	0402	
1	R2	560r	±10%	0402	
1	C11	1000pF	±10%	0402	
1	C12	15nF	±10%	0402	
1	C13	470pF	±10%	0402	

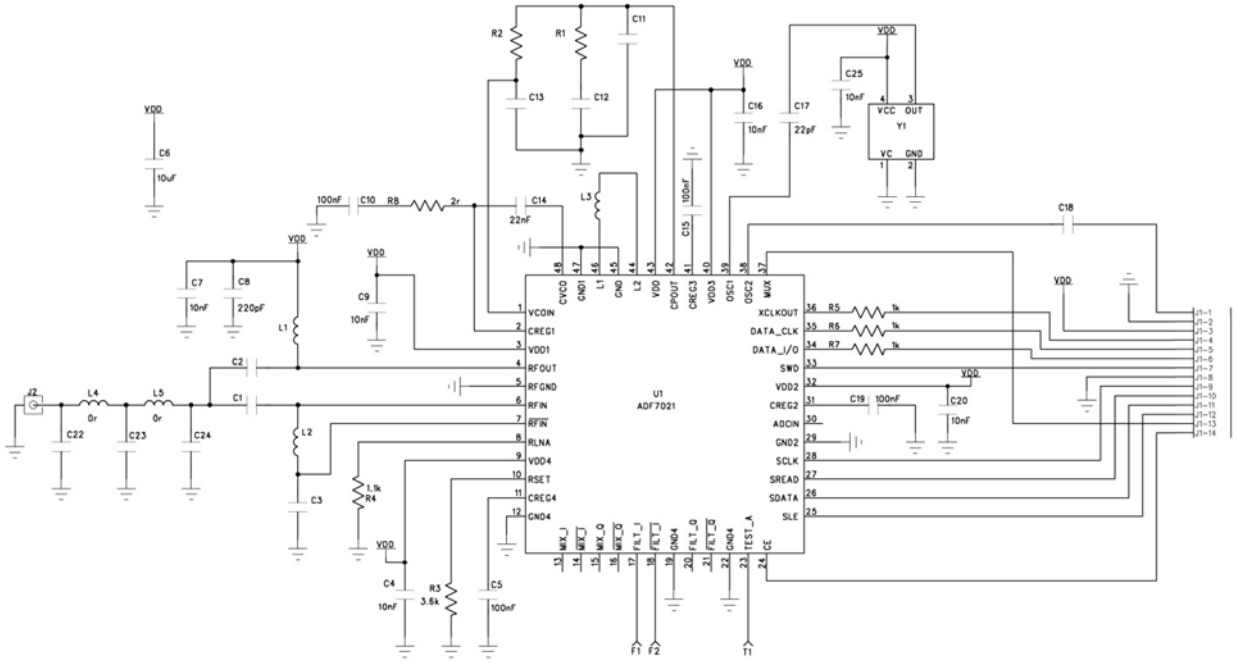
Pozn: místo TCXO použít Xtal 19.68MHz a dále místo SMD 0402 lze použít 0805

Pásmo 144 MHz (hodnoty z diplomové práce J.Bohátka: FM vysílač telemetrických dat APRS v pásmu 144MHz):

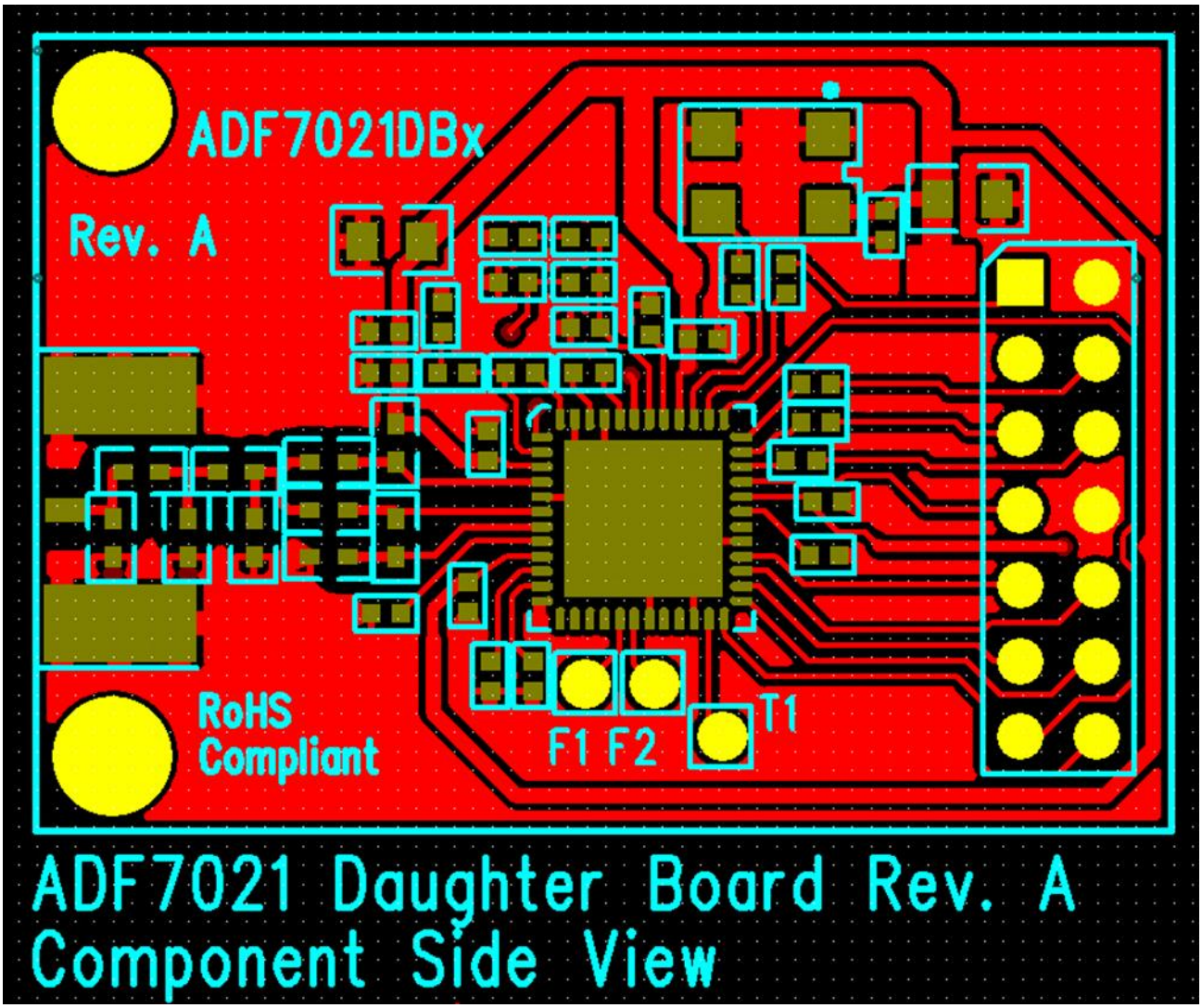
C.2 Modul vysílače

Part	Value	Package	Part	Value	Package
ADF7021	ADF7021	LFCSP_VQ-48	C53	33pF	C0805K
C1	12pF	C0805K	JP1	JP1	jumper
C2	18pF	C0805K	L1	100nH	L0805
C3	10pF	C0805K	L2	100nH	L0805
C4	100nF	C0805K	L3	100nH	L0805
C5	100nF	C0805K	L4	150nH	L0805
C7	10nF	C0805K	L5	130nH	L0805
C8	10nF	C0805K	LE33	LE33	SO-08
C9	10nF	C0805K	LED1	CHIP-LED0805	led
C10	100nF	C0805K	Q1	11,0592MHz	SM49
C11	1nF	C0805K	R1	390R	R0805
C12	15nF	C0805K	R2	1K0	R0805
C13	470pF	C0805K	R3	3K6	R0805
C14	22nF	C0805K	R4	1K1	R0805
C15	100nF	C0805K	R5	1K0	R0805
C16	100nF	C0805K	R6	1K0	R0805
C19	100nF	C0805K	R7	1K0	R0805
C20	100nF	C0805K	R8	3R9	R0805
C22	15pF	C0805K	R50	10K	R0805
C23	15pF	C0805K	R51	560R	R0805
C24	15pF	C0805K	SV1	L10P	con-amp-mt
C50	100nF	C0805K	SV2	L10P	con-amp-mt
C51	100nF	C0805K	X2	BU-SMA-G	con-coax
C52	33pF	C0805K			

Dále uvádím zapojení destičky EVAL-ADF7021-NDBZ5, kterou jsem se inspiroval:



A rovněž její PCB:


ADF7021DBx
Rev. A
RoHS
Compliant
F1 F2
T1
ADF7021 Daughter Board Rev. A
Component Side View

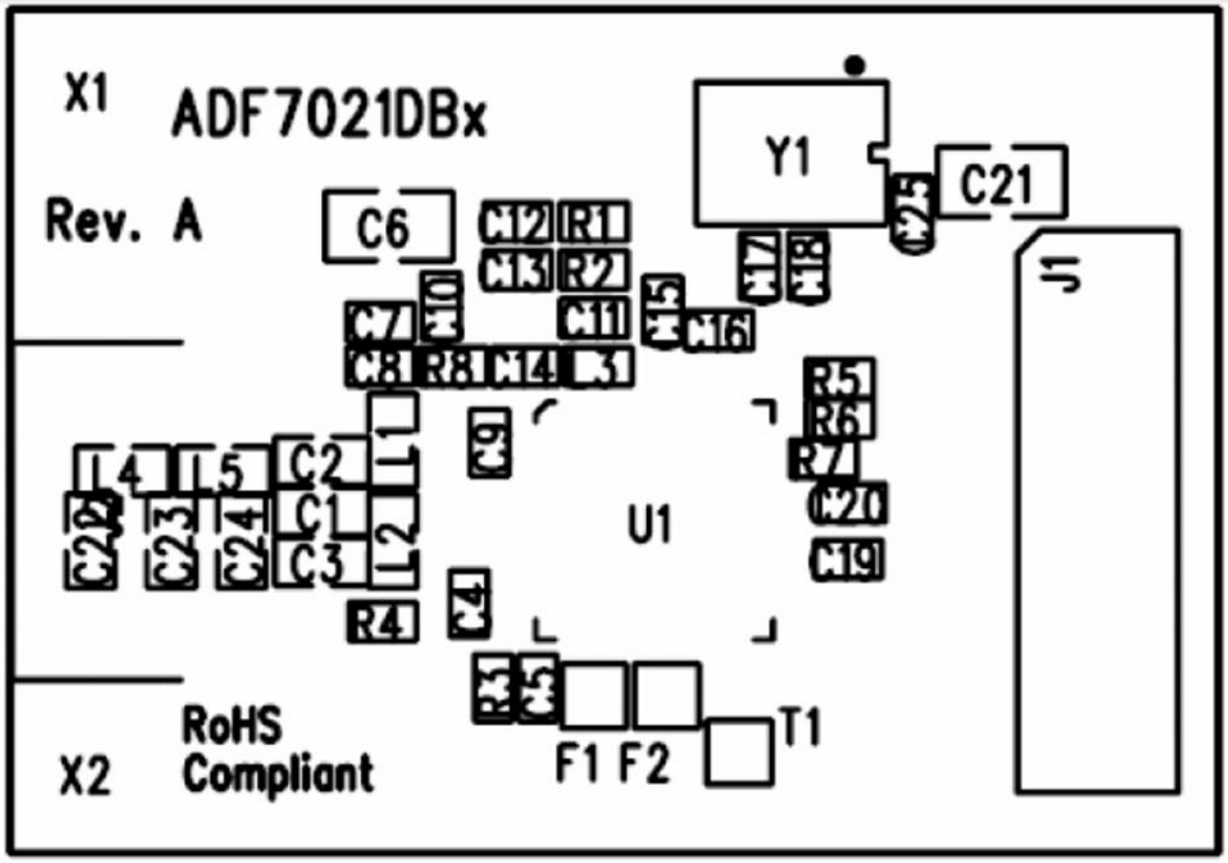
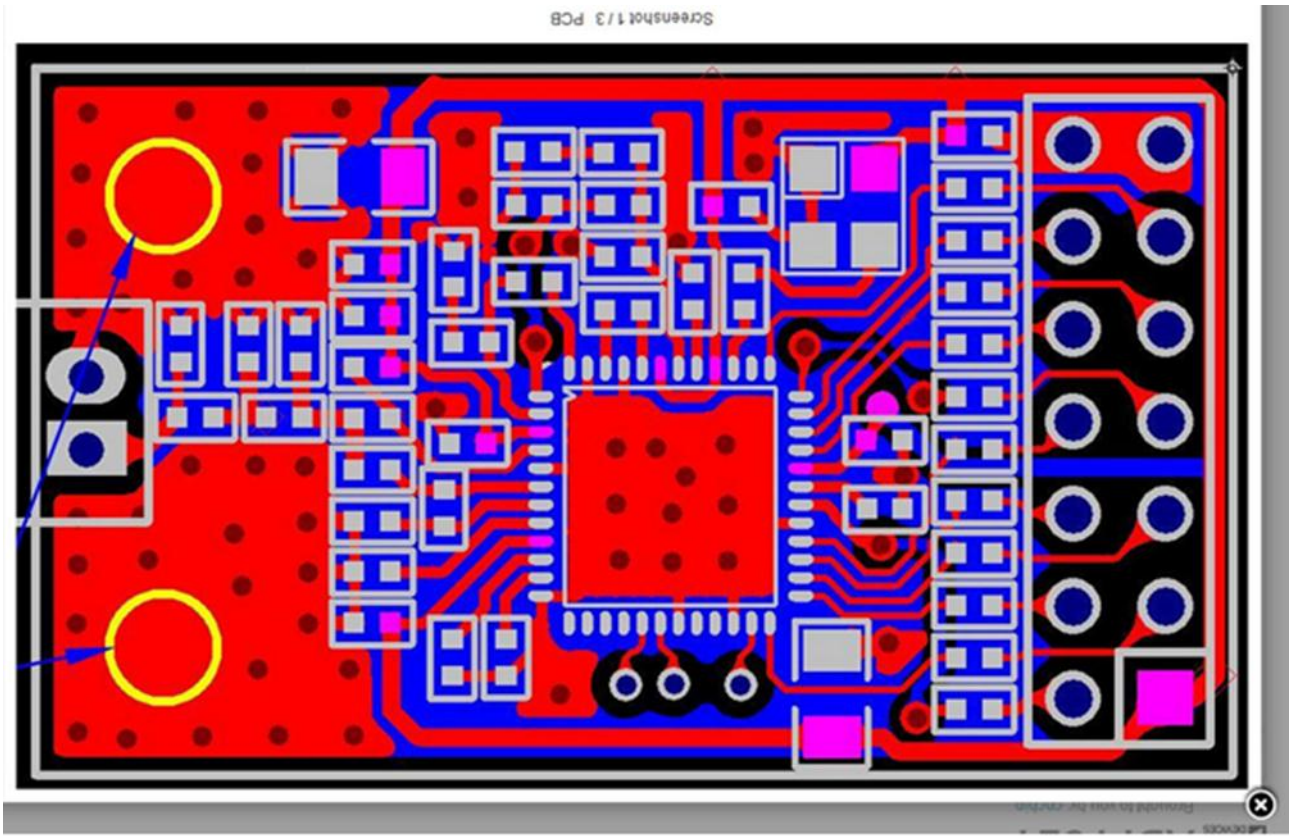
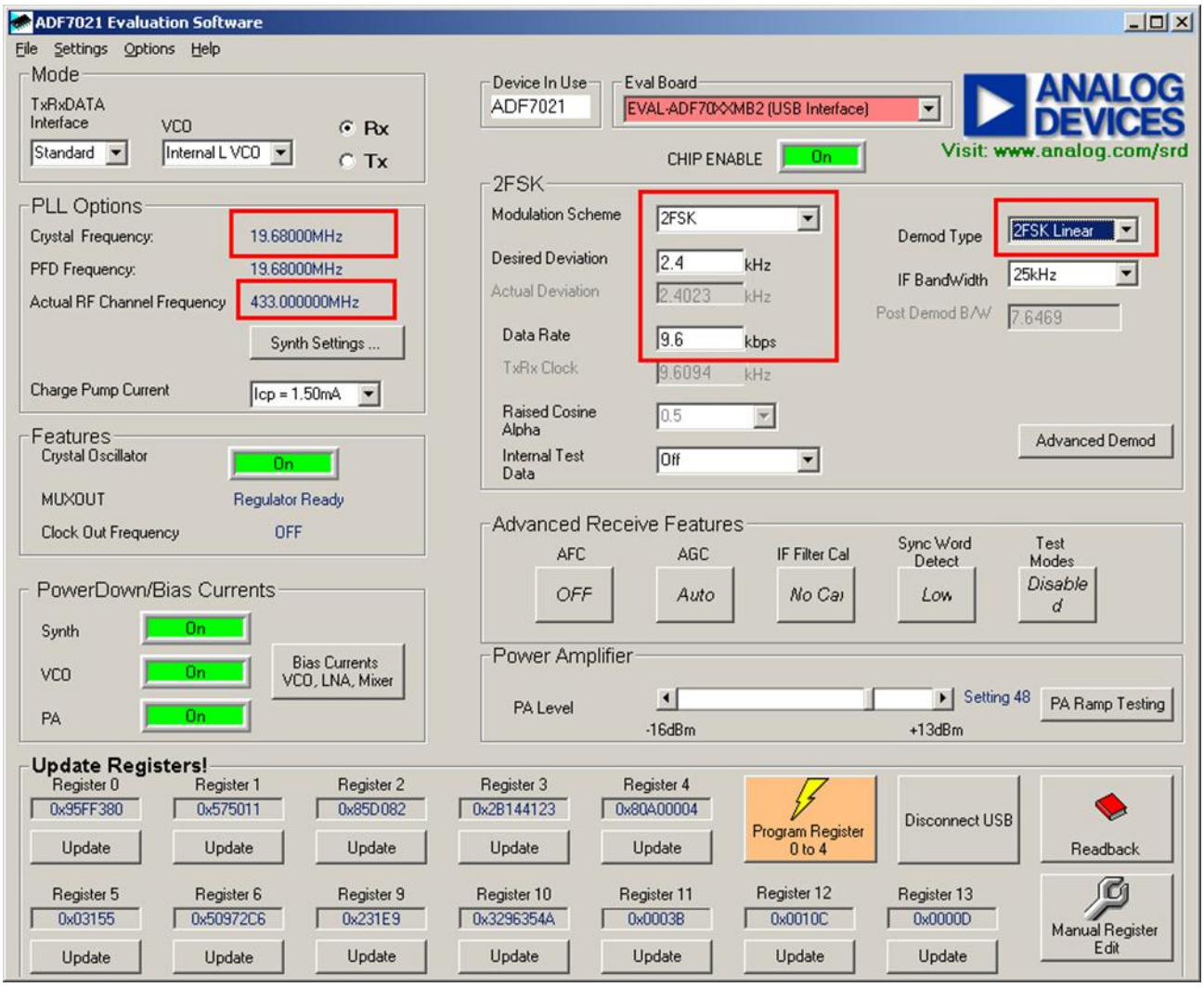


Figure 1. EVAL-ADF7021DBX silkscreen

To, že sloužila jako inspirace pro další vývojáře je zřejmé z následujícího PCB, rovněž objeveného v hlubinách internetu:



Mírně se liší motivem PCB a dále tím, že navíc obsahuje ochranné odpory mezi piny ADF7021 a konektorem.
K Evaluation Boardu dodává AD zdarma sw:



Tento sw dřívějších verzí můžeme ovšem použít, ovšem pokud použijeme mezi PC a naší deskou zapojení, obdobné Evaluation Boardu, jehož zapojení najdeme na webu AD. Záměrně píši starší verzi, čímž myslím verze sw používané s Evaluation Boardem, který se k PC připojoval pomocí paralelního portu.
Stávající Evaluation Boardy se připojují pomocí usb a výroba jejich ekvivalentů již není tak triviální. Proto je přirozené použít vlastní MCU (ATMega) k řízení destičky s ADF7021. V následujícím odstavci 5.3.1. jde o sw z diplomové práce Jana Bohatky FM vysílač APRS telemetrických dat v pásmu 144 MHz (VUT Brno 2011).

4.3.1 Vysílač s ATMega168

Nejprve si ukážeme příklad vysílače s daty přenášenými pomocí protokolu AX25 – viz 1.díl skript [1]. V podstatě půjde o obdobu vysílače pro CanSAT od PrattHobies. Vf část bude s ADF7021 dle předchozích odstavců v kap. 5.3. Jako MCU bude opět ATMega firmy ATMEL. Nejprve uvedeme firmware pro tento MCU.

Soubor **main.c**

```
#include "main.h"
#include "ADF7021.h"
```

```
char message;
unsigned int Time1=0;
unsigned int Time2=0;
long int SetByte;
char MAStart=0;
char MAset=TRUE;
char MAd;
int MAe;
char MAa=0;
char MAh;

char buffer2[10];

extern char Destination[21];
extern char Source[7];
extern char Comment[21];
extern int Adc1, Adc2, Adc3;
extern int ADTemp;
extern int txDelay;
extern char NDestinat;
extern int Adc0a, Adc1a,Adc2a,Adc3a,Adc4a;
extern int Adc0b, Adc1b,Adc2b,Adc3b,Adc4b;
extern int Adc0c, Adc1c,Adc2c,Adc3c,Adc4c;
//*****************************************************************************
//* Functions
//*****************************************************************************
extern void u_puts0();
extern void u_puts1();
extern void u_putc1();

extern void ADF7021_set();
extern void config_use_defaults();

extern void Clock();
extern void adc();
extern void adc_init();
extern void ax25sendHeader(void);
extern void Telemetry_Rep(void);
extern void Position(void);
extern void message_init(void);

extern void NMEA();
void MCU_init(void); /* Device initialization function declaration */
void Convert(void);
//*****************************************************************************
int main(){
  TIMSK1 |= (1<<TOIE1);
  TCCR1B |= (1<<CS12) | (0<<CS11) | (0<<CS10);  //ma byt TCCR0=0b00000011 CLK/256

  TCNT1H = CTH;
  TCNT1L = CTL;

  USART_Init();
  adc_init();
  DDRB = 0b11111111;       //0b11101100        //1-out 0-in
  DDRC = 0b11111111;       //0b11101100        //1-out 0-in

  sei();
  mSecondas_clo = 0;

  config_use_defaults();                       //Use standard register configuration
as defined
```

```
  NMEA();
  message_init();
  TimePer1 =2;
  TimePer2 =4;
  Comment[1]=' ';
  adc();
  NDestinat=1;
  MAd=0;

  while(1)
       {
       if (mSecondas_clo >= 100)
              {
       Secondas_clo++;
       mSecondas_clo = mSecondas_clo-100;
              Time1++;
       Clock();
              MAStart=1;
              }

/*     if (MAset == TRUE)
              {
              ADF7021_set();
              MAset=FALSE;
              }
*/
       if (MAStart ==1)
              {
              u_puts1("\r\n");
/*            if (Hours_clo <10 ) u_putc1('0');
              itoa((Hours_clo),tempstr,10);     // číslo int1 převeď na string tempstr,
              u_puts1(tempstr);           // a zapiš do UART
              u_putc1(':');
              if (Minutes_clo <10 ) u_putc1('0');
              itoa(Minutes_clo,tempstr,10);     // číslo int1 převeď na string tempstr,
              u_puts1(tempstr);           // a zapiš do UART
              u_putc1(':');
              if (Secondas_clo <10 ) u_putc1('0');
              itoa(Secondas_clo,tempstr,10);     // číslo int1 převeď na string
tempstr,
              u_puts1(tempstr);           // a zapiš do UART

              u_puts1(" Adc0a ");
              itoa(Adc0a,tempstr,10);     // číslo int1 převeď na string tempstr,
              u_puts1(tempstr);           // a zapiš do UART

              u_puts1(" Adc0b ");
              itoa(Adc0b,tempstr,10);     // číslo int1 převeď na string tempstr,
              u_puts1(tempstr);           // a zapiš do UART

              u_puts1(" Adc0c ");
              itoa(Adc0c,tempstr,10);     // číslo int1 převeď na string tempstr,
              u_puts1(tempstr);           // a zapiš do UART
*/
              if (((Time2/TimePer1)*TimePer1)==Time2){
                           ADF7021_set();
                     //     _delay_us(2);
                           ax25sendHeader();
       // Begin transmission of packet destination address
                           Position();
                           u_putc1('P');
```

```
                    //      _delay_us(2);
                            dd_adf7020_chip_sel(FALSE);
                            }
              if (((Time2/TimePer2)*TimePer2)==Time2){
                            ADF7021_set();
                    //      _delay_us(2);
                            adc();
                            ax25sendHeader();
       // Begin transmission of packet destination address
                            Telemetry_Rep();
                            u_putc1('A');
                    //      _delay_us(2);
                            dd_adf7020_chip_sel(FALSE);
                            }
              }
       Time2=Time1;
       MAStart=0;
       }
  }

//*****************************************************************************
void Uart_set(){
char MAc=0;
switch(Temp1){
       case '$' :
              bufferindex1 =0;
              SetBits=0;
              memset(buffer1,' ',11);
              Time1=0;
              Time2=0;
              MAStart=0;
              break;
       case ' ':
              bufferindex1 =0;
              if ((buffer1[1]=='C') && (buffer1[2]=='L') && (buffer1[3]=='O'))
SetBits=1;
              if ((buffer1[1]=='F') && (buffer1[2]=='R') && (buffer1[3]=='A'))
SetBits=2;
              if ((buffer1[1]=='I') && (buffer1[2]=='N') && (buffer1[3]=='T'))
SetBits=3;
              if ((buffer1[1]=='P') && (buffer1[2]=='O') && (buffer1[3]=='W'))
SetBits=5;
              if ((buffer1[1]=='S') && (buffer1[2]=='E') && (buffer1[3]=='T'))
SetBits=6;
              if ((buffer1[1]=='P') && (buffer1[2]=='E') && (buffer1[3]=='1'))
SetBits=4;
              if ((buffer1[1]=='P') && (buffer1[2]=='E') && (buffer1[3]=='2'))
SetBits=7;
              if ((buffer1[1]=='S') && (buffer1[2]=='T') && (buffer1[3]=='A'))
SetBits=8;

              if ((buffer1[1]=='S') && (buffer1[2]=='O') && (buffer1[3]=='U'))
SetBits=20;
              if ((buffer1[1]=='D') && (buffer1[2]=='E') && (buffer1[3]=='S'))
SetBits=21;
              if ((buffer1[1]=='C') && (buffer1[2]=='O') && (buffer1[3]=='M'))
SetBits=22;
              if ((buffer1[1]=='A') && (buffer1[2]=='D') && (buffer1[3]=='0'))
SetBits=23;
              if ((buffer1[1]=='A') && (buffer1[2]=='D') && (buffer1[3]=='1'))
SetBits=24;
```

```
              if ((buffer1[1]=='A') && (buffer1[2]=='D') && (buffer1[3]=='2'))
SetBits=25;
              if ((buffer1[1]=='A') && (buffer1[2]=='D') && (buffer1[3]=='3'))
SetBits=26;
              if ((buffer1[1]=='A') && (buffer1[2]=='D') && (buffer1[3]=='4'))
SetBits=27;
              memset(buffer1,' ',11);
              break;
       case ';':
              switch (SetBits)
                     {
                     case 1:
                            r1_tx_L = (r1_tx_L & 0b1111100001111111);
       //set Clkout_Divide
                            r1_tx_L = (r1_tx_L + (SetByte << 7));
                            u_puts1(" CLK");
                            break;
                     case 2:
                            r0_tx_L = (r0_tx_L & 0b0000000000001111);
       //set Fractional_N
                            r0_tx_L = (r0_tx_L + (SetByte << 4));

                            r0_tx_H = (r0_tx_H & 0b1111111111111000);
                            r0_tx_H = (r0_tx_H + (SetByte >> 12));
                            u_puts1(" FRA");
                            break;
                     case 3:
                            r0_tx_H = (r0_tx_H & 0b1111100000000111);
       //set Integer_N
                            r0_tx_H = (r0_tx_H + (SetByte << 3));
                            u_puts1(" INT");
                            break;
                     case 4:
                            TimePer1 = SetByte;
                            if (SetByte <2) TimePer1 =2;
                            u_puts1(" PE1");
                            break;
                     case 5:
                            r2_tx_L = (r2_tx_L & 0b0001111111111111);
       //set Power_Amplifier
                            r2_tx_L = (r2_tx_L + ((SetByte & 0b000111) << 13));

                            r2_tx_H = (r2_tx_H & 0b1111111111111000);
                            r2_tx_H = (r2_tx_H + ((SetByte & 0b111000) >> 3));
                            u_puts1(" POW");
                            break;
                     case 6:
                            MAset=TRUE;
                            u_puts1(" SET");
                            break;
                     case 7:
                            TimePer2 = SetByte;
                            if (SetByte <2) TimePer2 =2;
                            u_puts1(" PE2");
                            break;
                     case 8:
                            txDelay = SetByte;
                            if (txDelay <2) txDelay =2;
                            u_puts1(" STA");
                            break;
                     case 20:
                            for (MAa=1;MAa<7;MAa++)
```

```
                        {Source[MAa]=buffer1[MAa];
                        }
                u_puts1(" SOU");
                break;
        case 21:
                memset(Destination,' ',21);
                for (MAa=1;MAa<bufferindex1;MAa++)
                        {
                        if (buffer1[MAa]==',')
                                NDestinat++;
                                else
                                        Destination[MAa]=buffer1[MAa];
                        }
                u_puts1(" DES");
                break;
        case 22:
                for (MAa=1;MAa<=bufferindex1;MAa++)
                        {Comment[MAa]=buffer1[MAa];}
                Comment[MAa]=' ';
                u_puts1(" COM");
                break;
        case 23:
                for (MAa=1;MAa<=bufferindex1;MAa++)
                        {
                        if (buffer1[MAa]==',')
                                {
                                MAc++;
                                switch (MAc)
                                        {
                                        case 1 :
                                                Adc0a=MAe;
                                                break;
                                        case 2 :
                                                Adc0b=MAe;
                                                break;
                                        default :
                                                Adc0c=MAe;
                                                MAc=0;
                                                break;
                                        }
                                MAd=0;

                                }
                                else
                                        {
                                        Convert();
                                        }
                        }
                u_puts1(" AD0");
                break;
        case 24:
                for (MAa=1;MAa<=bufferindex1;MAa++)
                        {
        //              if (buffer1[MAa]=='-') MAh=1;
                        if (buffer1[MAa]==',')
                                {
                                MAc++;
                                switch (MAc)
                                        {
                                        case 1 :
                                                Adc1a=MAe;
                                                break;
```

```
                                        case 2 :
                                               Adc1b=MAe;
                                               break;
                                        default :
                                               Adc1c=MAe;
                                               MAc=1;
                                               break;
                                        }
                                 MAd=0;

                                 }
                                 else
                                        {
                                        Convert();
                                        }
                          }
                   u_puts1(" AD1");
                   break;
            case 25:
                   for (MAa=1;MAa<=bufferindex1;MAa++)
                          {
                          if (buffer1[MAa]==',')
                                 {
                                 MAc++;
                                 switch (MAc)
                                        {
                                        case 1 :
                                               Adc2a=MAe;
                                               break;
                                        case 2 :
                                               Adc2b=MAe;
                                               break;
                                        default :
                                               Adc2c=MAe;
                                               MAc=1;
                                               break;
                                        }
                                 MAd=0;

                                 }
                                 else
                                        {
                                        Convert();
                                        }
                          }
                   u_puts1(" AD2");
                   break;
            case 26:
                   for (MAa=1;MAa<=bufferindex1;MAa++)
                          {
                          if (buffer1[MAa]==',')
                                 {
                                 MAc++;
                                 switch (MAc)
                                        {
                                        case 1 :
                                               Adc3a=MAe;
                                               break;
                                        case 2 :
                                               Adc3b=MAe;
                                               break;
                                        default :
```

```
                                                Adc3c=MAe;
                                                MAc=1;
                                                break;
                                        }
                                MAd=0;

                                }
                                else
                                        {
                                        Convert();
                                        }
                                }
                        u_puts1(" AD3");
                        break;
                case 27:
                        for (MAa=1;MAa<=bufferindex1;MAa++)
                                {
                                if (buffer1[MAa]==',')
                                        {
                                        MAc++;
                                        switch (MAc)
                                                {
                                                case 1 :
                                                        Adc4a=MAe;
                                                        break;
                                                case 2 :
                                                        Adc4b=MAe;
                                                        break;
                                                default :
                                                        Adc4c=MAe;
                                                        MAc=1;
                                                        break;
                                                }
                                        MAd=0;

                                        }
                                        else
                                                {
                                                Convert();
                                                }
                                }
                        u_puts1(" AD4");
                        break;
                default : u_puts1("chyba");break;
                }
        SetBits=0;
        break;
    default :
        bufferindex1++;
        buffer1[bufferindex1]=Temp1;
        if (SetBits <=19) SetByte=(atoi(buffer1));
        break;
    }
}

//*****************************************************************************
void Convert(void){
    MAd++;
    buffer2[MAd]=buffer1[MAa];
//  MAe=(atoi(buffer2));
    MAe = (((buffer2[1]-48)*1000)+((buffer2[2]-48)*100)+((buffer2[3]-
48)*10)+((buffer2[4])-48));
```

```
//      if (MAh==1) MAe=0-MAe;
//      MAh=0;
        }

//*****************************************************************************
//Interrupts
//*****************************************************************************
ISR(TIMER1_OVF_vect)
  {
        mSecondas_clo = (mSecondas_clo + 100);
        TCNT1H = CTH;
        TCNT1L = CTL;
  }

ISR(USART0_RXC_vect)
  {
        Temp=UDR0;
        NMEA();
  }

ISR(USART1_RXC_vect)
  {
        Temp1=UDR1;
        Uart_set();
  }
```

Soubor **main.h**

```
#ifndef _main_H_
#define _main_H_

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/iomxx4.h>          // definice io registrů atd
#include <stdio.h>                       // Standard IO facilities
#include <inttypes.h>                    // Integer Type conversions
#include <util/delay.h>                  // Busy-wait delay loops
#include <avr/sleep.h>

#define TRUE 1
#define FALSE 0

//*****************************************************************************
//* Global variables
//*****************************************************************************
unsigned char mSecondas_clo, Secondas_clo, Minutes_clo, Hours_clo, Days_clo,
Months_clo, Years_clo;
unsigned int tempstr[10];
unsigned char bufferindex1,SetBits;
char buffer1[25];

int Temp,Temp1,TimePer1,TimePer2;

unsigned int TmpTime1;
unsigned int TmpTime2;

extern int Course_gps, Speed_gps, Altitude_gps;
```

```
extern int LongitudeGps, LongitudeM_gps;

extern void USART_Init();

//*****************************************************************************
//* Macro
//*****************************************************************************
#define CTH 0x57                  //CTO=22336-0x5740=2^16 - (1/(256/11059200))
#define CTL 0x40

#define F_CPU 11059200


#endif /* _main_H_ */
```

Soubor **Message.c**

```
extern char LatitudeP_gps,LongitudeP_gps;
extern unsigned char Secondas_clo, Minutes_clo, Hours_clo, Days_clo, Months_clo,
Years_clo;
extern unsigned int tempstr[10];
extern int LatitudeGps,LatitudeM_gps;
extern int LongitudeGps,LongitudeM_gps;
extern int Course_gps, Speed_gps, Altitude_gps;
extern int Adc0 , Adc1 ,Adc2 ,Adc3 ,Adc4 ,Adc6 ;
int Adc0a, Adc1a,Adc2a,Adc3a,Adc4a;
int Adc0b, Adc1b,Adc2b,Adc3b,Adc4b;
int Adc0c, Adc1c,Adc2c,Adc3c,Adc4c;
char Source[7];
char Destination[21];
char Comment[21];
char NDestinat;

void Telemetry_Rep(void);
void Position(void);
void Adress_field(void);
void message_init(void);
void Adress_Destination(void);

extern void m_puts(unsigned char *text2);

//*****************************************************************************
void Position()
      //@092345z4903.50N/07201.75W>088/036
  {
      Adress_field();
      //APRS101.pdf page33

      m_puts("@");
      if ((Hours_clo) < 10) ax25sendByte('0'); // Send the time
      itoa((Hours_clo),tempstr,10);
      m_puts(tempstr);
      if (Minutes_clo < 10) ax25sendByte('0');
      itoa(Minutes_clo,tempstr,10);
      m_puts(tempstr);
      if (Secondas_clo < 10) ax25sendByte('0');
      itoa(Secondas_clo,tempstr,10);
      m_puts(tempstr);
```

```
      m_puts("h");                                                  // Tag it as zulu
time

      if (LatitudeGps <1000) ax25sendByte('0');       // Send the Latitude
      if (LatitudeGps <100) ax25sendByte('0');
      if (LatitudeGps <10) ax25sendByte('0');
      itoa(LatitudeGps,tempstr,10);
      m_puts(tempstr);
      ax25sendByte('.');
      if (LatitudeM_gps <10) ax25sendByte('0');
      itoa(LatitudeM_gps,tempstr,10);
      m_puts(tempstr);
      ax25sendByte(LatitudeP_gps);
      ax25sendByte('/');

      if (LongitudeGps <10000) ax25sendByte('0');     // Send the Longitude
      if (LongitudeGps <1000) ax25sendByte('0');
      if (LongitudeGps <100) ax25sendByte('0');
      if (LongitudeGps <10) ax25sendByte('0');
      itoa(LongitudeGps,tempstr,10);
      m_puts(tempstr);
      ax25sendByte('.');
      if (LongitudeM_gps <10) ax25sendByte('0');
      itoa(LongitudeM_gps,tempstr,10);
      m_puts(tempstr);
      ax25sendByte(LongitudeP_gps);
      ax25sendByte('>');

      if (Course_gps <100) ax25sendByte('0');         // Send the Course
      if (Course_gps <10) ax25sendByte('0');
      itoa(Course_gps,tempstr,10);
      m_puts(tempstr);
      ax25sendByte('/');

      if (Speed_gps <100) ax25sendByte('0');          // Send the Speed
      if (Speed_gps <10) ax25sendByte('0');
      itoa(Speed_gps,tempstr,10);
      m_puts(tempstr);

      ax25sendByte('/');
      m_puts("A=");
      itoa(Altitude_gps,tempstr,10);
      m_puts(tempstr);

      ax25sendByte('/');
      comment();

      ax25sendCrc();
      ax25sendBye();
  }

//*************************************************************************
void Telemetry_Rep(void)
      //T#MIC199,000,255,073,123,01101001
  {
      //APRS101.pdf page68
      unsigned char TempTelemetry;

      Adress_field();
      m_puts("T#MIC");

      itoa(Adc0,tempstr,10);
```

```
	m_puts(tempstr);
	ax25sendByte(',');

	itoa(Adc1,tempstr,10);
	m_puts(tempstr);
	ax25sendByte(',');

	itoa(Adc2,tempstr,10);
	m_puts(tempstr);
	ax25sendByte(',');

	itoa(Adc3,tempstr,10);
	m_puts(tempstr);
	ax25sendByte(',');

	itoa(Adc4,tempstr,10);
	m_puts(tempstr);
	ax25sendByte(',');

	TempTelemetry=Adc6;
	for (char eee=1;eee<8;eee++)
		{
		TempTelemetry= (Adc6 & 0b00000001);
		if (TempTelemetry ==1) ax25sendByte('1');
			else ax25sendByte('0');
		Adc6=(Adc6>>1);
		}

	ax25sendByte('/');
	comment();

	ax25sendCrc();

	Adress_field();
	ax25sendByte(':');
	ax25sendByte(Destination[1]);
	ax25sendByte(Destination[2]);
	ax25sendByte(Destination[3]);
	ax25sendByte(Destination[4]);
	ax25sendByte(Destination[5]);
	ax25sendByte(Destination[6]);
	m_puts("  :");


	m_puts("PARM.Voltage,");
	m_puts("Temp1,");
	m_puts("Temp2,");
	m_puts(" ,");
	m_puts(" ,");
	m_puts(" ,");
	m_puts(" ,");
	m_puts(" ,");
	m_puts(" ,");
	m_puts(" ");
	ax25sendCrc();

	Adress_field();
	ax25sendByte(':');
	ax25sendByte(Destination[1]);
	ax25sendByte(Destination[2]);
	ax25sendByte(Destination[3]);
	ax25sendByte(Destination[4]);
```

```
        ax25sendByte(Destination[5]);
        ax25sendByte(Destination[6]);
        m_puts("  :");

        m_puts("EQNS,");

        itoa(Adc0a,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');
        itoa(Adc0b,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');
        itoa(Adc0c,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');

        itoa(Adc1a,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');
        itoa(Adc1b,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');
        itoa(Adc1c,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');

        itoa(Adc2a,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');
        itoa(Adc2b,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');
        itoa(Adc2c,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');

        itoa(Adc3a,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');
        itoa(Adc3b,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');
        itoa(Adc3c,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');

        itoa(Adc4a,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');
        itoa(Adc4b,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');
        itoa(Adc4c,tempstr,10);
        m_puts(tempstr);
        ax25sendByte(',');

        ax25sendCrc();
        ax25sendBye();
        }

//*************************************************************************
void Adress_field(void){
```

```
       ax25sendByte(0x7E);
       ax25address(Source[1]);                                    // A
       ax25address(Source[2]);                                    // P
       ax25address(Source[3]);                                    // A
       ax25address(Source[4]);                                    // V
       ax25address(Source[5]);                                    // R
       ax25address(Source[6]);                                    // 0
       ax25sendByte(0x60);                                  // SSID=0

       Adress_Destination();

       // Finish out the header with two more bytes
       ax25sendByte(0x03);                                  // Control field - 0x03
is APRS UI-frame
       ax25sendByte(0xF0);                                  // Protocol ID - 0xF0 is
no layer 3

}


//*****************************************************************************
void message_init(void){
       Source[1]='A';
       Source[2]='H';
       Source[3]='O';
       Source[4]='J';
       Source[5]='5';
       Source[6]='6';

       Destination[1]='1';
       Destination[2]='2';
       Destination[3]='3';
       Destination[4]='4';
       Destination[5]='5';
       Destination[6]='6';

       Destination[7]=' ';
       Destination[8]=' ';
       Destination[9]=' ';
       Destination[10]=' ';
       Destination[11]=' ';
       Destination[12]=' ';
       }

//*****************************************************************************
void comment(void){
       char Maa=1;
       while (!(Comment[Maa]==' '))
              {
              ax25sendByte(Comment[Maa]);
              Maa++;
              }
       }

//*****************************************************************************
void Adress_Destination(void){
       for (char MEa=(NDestinat*6);MEa>6;MEa=MEa-6)
              {
              // Begin transmission of packet Destination address
              ax25address(Destination[(MEa-5)]);
       // A
```

```
              ax25address(Destination[(MEa-4)]);
      // P
              ax25address(Destination[(MEa-3)]);
      // A
              ax25address(Destination[(MEa-2)]);
      // V
              ax25address(Destination[(MEa-1)]);
      // R
              ax25address(Destination[MEa]);                                  // 0
              ax25sendByte(0x60);                                  // Station ID (11)
              }

      ax25address(Destination[1]);                                        // A
      ax25address(Destination[2]);                                        // P
      ax25address(Destination[3]);                                        // A
      ax25address(Destination[4]);                                        // V
      ax25address(Destination[5]);                                        // R
      ax25address(Destination[6]);                                        // 0
      ax25sendByte(0x65);                                  // Station ID (11)
      }
```

Soubor **Uart.c**

```
#include "main.h"

#define BAUDRATE0 4800            //set desired baud rate
#define UBRRVAL0 ((F_CPU/(BAUDRATE0*16UL))-1)
#define BAUDRATE1 115200   //set desired baud rate
#define UBRRVAL1 ((F_CPU/(BAUDRATE1*16UL))-1)

//*****************************************************************************
void USART_Init(){
      //Set baud rate
      UBRR0L=UBRRVAL0;           //low byte
      UBRR0H=(UBRRVAL0>>8);      //high byte

      //Enable Transmitter and Receiver and Interrupt on receive complete
      UCSR0B=((1<<RXCIE0)|(1<<TXEN0)|(1<<RXEN0));
      //Set data frame format: asynchronous mode,no parity, 1 stop bit, 8 bit size
      UCSR0C=(0<<UMSEL01)|(0<<UMSEL00)|(0<<UPM01)|(0<<UPM00)|(0<<USBS0)|(1<<UCSZ01)|(1
<<UCSZ00)|(0<<UCPOL0);

      //Set baud rate
      UBRR1L=UBRRVAL1;           //low byte
      UBRR1H=(UBRRVAL1>>8);      //high byte
      UCSR1A=0x00;
      //Enable Transmitter and Receiver and Interrupt on receive complete
      UCSR1B=((1<<RXCIE1)|(1<<TXEN1)|(1<<RXEN1));
      //Set data frame format: asynchronous mode,no parity, 1 stop bit, 8 bit size
      UCSR1C=(0<<UMSEL11)|(0<<UMSEL10)|(0<<UPM11)|(0<<UPM10)|(0<<USBS1)|(1<<UCSZ11)|(1
<<UCSZ10)|(0<<UCPOL0);
  }

//*****************************************************************************
// UART pošli znak:
void u_putc0( unsigned char data0 ){
      while ( !( UCSR0A & (1<<UDRE0)) )     // Wait for empty transmit buffer
      ;
      UDR0 = data0;                         // Put data into buffer, sends the data
  }
```

```
void u_putc1( unsigned char data1 ){
       while ( !( UCSR1A & (1<<UDRE1)) )     // Wait for empty transmit buffer
       ;
       UDR1 = data1;                         // Put data into buffer, sends the data
  }

//***************************************************************************
// UART pošli string:
void u_puts0(unsigned char *text0 ){
       unsigned char i=0,temp0;
       do
       {
              temp0 = text0[i];
              if(temp0==0) break;
              u_putc0(temp0);
              i++;
       }
       while(temp0>0);
  }

void u_puts1(unsigned char *text1 ){
       unsigned char ii=0,temp1;
       do
       {
              temp1 = text1[ii];
              if(temp1==0) break;
              u_putc1(temp1);
              ii++;
       }
       while(temp1>0);
}

/*
//***************************************************************************
// UART přijmi znak:
unsigned char u_getc0( void ){
       while ( !(UCSR0A & (1<<RXC0)) )       // Wait for data to be received
       ;
       return UDR0;                          // Get and return received data from buffer
  }

unsigned char u_getc1( void ){
       while ( !(UCSR1A & (1<<RXC1)) )       // Wait for data to be received
       ;
       return UDR1;                          // Get and return received data from buffer
  }

*/
```

Soubor **Adc.c**

```
#include "main.h"
#include <inttypes.h>
#define ADC_VREF_TYPE 0x40

//***************************************************************************
char kanal;
//int pom1,pom2;
//int realne;
```

```
int Adc0, Adc1,Adc2,Adc3,Adc4,Adc5,Adc6;
//int ADTemp;

void adc(void);
void adc_init(void);

//*****************************************************************************
void adc_init(void){
       ADMUX = 0x40;        //extrerni Uref
       ADCSRA = 0x87;
       ADCSRB = 0x00;
       kanal=0;
       }

//*****************************************************************************
unsigned int read_adc(unsigned char kanal)
{
    ADMUX = ((ADMUX & 0xf0)+kanal);
    ADCSRA |= 0x40;                     //start
    while ((ADCSRA & 0x10)==0);
    ADCSRA |= 0x10;
    return ADCW;
}


//*****************************************************************************
void adc(void){
while (kanal <= 8)
       {
       switch(kanal){
              case 0:
                     Adc0 = read_adc(kanal);
                     break;
              case 1:
                     Adc1 = read_adc(kanal);
                     break;
              case 2:
                     Adc2 = read_adc(kanal);
                     break;
              case 3:
                     Adc3 = read_adc(kanal);
                     break;
              case 4:
                     Adc4 = read_adc(kanal);
                     break;
              default:
                     break;
              }
       kanal++;
       }
kanal=0;
}
```

Soubor **Clock.c**

```
#include "clock.h"
#include "main.h"

//*****************************************************************************
//Hodiny podle mikrokontroleru
```

```
//*****************************************************************************
void Clock(void){
  char Months_p;
  if (Secondas_clo >= 60){
    Minutes_clo++;
    Secondas_clo = 0;
       TmpTime1++;
    }
  if(Minutes_clo >= 60){
    Hours_clo++;
    Minutes_clo = 0;
    }
  if (Hours_clo >= 24){
    Days_clo++;
    Hours_clo = 0;
       TmpTime1 = ((Hours_clo * 60) + Minutes_clo);
       TmpTime2 = TmpTime1;
    }
 switch(Months_clo){
    case 2: Months_p = 28; break;
    case 4: Months_p = 30; break;
    case 6: Months_p = 30; break;
    case 9: Months_p = 30; break;
    case 11: Months_p = 30; break;
    default: Months_p = 31; break;
    }
 if (Days_clo > Months_p){
    Months_clo ++;
    Days_clo = 1;
    }
  if (Months_clo > 12){
    Years_clo ++;
    Months_clo = 1;
    if (Years_clo >= 100) Years_clo = 0;
    }
  }
```

Soubor **Clock.h**

```
#ifndef _clock_H_
#define _clock_H_


/************************** API Function Prototypes ***************************/
void Clock();

extern unsigned char mSecondas_clo, Secondas_clo, Minutes_clo, Hours_clo, Days_clo,
Months_clo,Years_clo;
extern unsigned int TmpTime1;
extern unsigned int TmpTime2;

//*****************************************************************************

#endif /* _clock_H_ */
```

Soubor **Ax25.c**

```
#include "main.h"
```

```
#define TxDxData    0b000111111

#define COUNTER_BIT        115          //=2^8-(827us/(TCCR0/11,0592Mhz))
#define COUNTER_1200 115          //=2^8-(833us/(TCCR2/11,0592Mhz))
#define COUNTER_2200 177          //=2^8-(454us/(TCCR2/11,0592Mhz))

int txDelay=50;                   // Number of 6.7ms delay cycles (send flags)
//*************************************************************************
// Variables
//*************************************************************************
unsigned char loop_delay;
static char Phase,Clk;
unsigned int Counter2;
static unsigned int crc;
extern char message_gps;

void DelayBit(void);
void ChangePhase(void);
void ax25sendHeader(void);
void ax25sendByte(unsigned char txbyte);
void ChangeFreq(void);
void ax25crcBit(int lsb_int);
void ax25address(unsigned char txbyteA);

//*************************************************************************
// Interrupt
//*************************************************************************
ISR (TIMER2_OVF_vect){
       TCNT2 = Counter2;
       ChangePhase();
       }

//*************************************************************************
// Functions
//*************************************************************************
void ax25sendHeader(void){
       crc = 0xFFFF;                                                  //Initialize
the crc register
       UCSR0B &= ~(1<<RXCIE0);                                        //RX0
Complete Interrupt Disable
//     UCSR1B &= ~(1<<RXCIE1);                                        //RX1
Complete Interrupt Disable

       DelayBit();                                                          //
Pause for the bit to be sent

       Phase=FALSE;
       Counter2=COUNTER_1200;

       TCNT2 = COUNTER_1200;
       TCCR2B |= ((0<<CS22) | (1<<CS21) | (1<<CS20));  //ma byt TCCR0 = 0b00000011
       CLK/64
       TIMSK2 |= (1<< TOIE2);
       //Timer/Counter2 Overflow Inerrupt Enable

       for (loop_delay = 0 ; loop_delay < txDelay ; loop_delay++)
              {
              (ax25sendByte(0x7E));
              }
}
```

```
//*****************************************************************************/
void ax25sendByte(unsigned char txbyte){
	char loop;
	static char bitbyte;
	static char bit_zero;
	static unsigned char sequential_ones;

	bitbyte = txbyte;                                          // Bitbyte will be
rotated through
	for (loop = 0 ; loop < 8 ; loop++)                    // Loop for eight bits in
the byte
		{
		bit_zero = bitbyte & 0x01;                    // Set aside the least
significant bit
		if (txbyte == 0x7E)                                  // Is the transmit
character a flag?
			{
			sequential_ones = 0;                  // it is immune from
sequential 1's
			}
			else                                                  // The
transmit character is not a flag
				{
				(ax25crcBit(bit_zero));               // So modify the
checksum
				}

		if (!(bit_zero))                                     // Is the least
significant bit low?
			{
			sequential_ones = 0;                  // Clear the number of
ones we have sent
			ChangeFreq();
			}
			else                                                  // Else,
least significant bit is high
				{
				if (++sequential_ones >= 5)   // Is this the 5th "1" in
a row?
					{
					DelayBit();                           // Go ahead
and send it
					ChangeFreq();
					sequential_ones = 0;  // Clear the number of
ones we have sent
					}
				}
		bitbyte >>= 1;                                       // Shift the
reference byte one bit right
		DelayBit();                                          // Pause for
the bit to be sent
	}
}


//*****************************************************************************
void DelayBit(void){
	TCNT0 = COUNTER_BIT;
	TCCR0B |= ((0<<CS02) | (1<<CS01) | (1<<CS00));  //ma byt TCCR0=0b00000011
	CLK/64

	loop_until_bit_is_set(TIFR0,0);
```

```
	TIFR0 = TIFR0 & 0b00000001;
	TCCR0B |= ((0<<CS02) | (0<<CS01) | (0<<CS00));  //off
	}

//*****************************************************************************
void ax25sendBye(void){

	ax25sendByte(0x7E);                                    // Send a flag to end the
packet

	TCCR0B = 0x00;                                                 //Timer/Counter
stopped
	TIMSK2 = 0x00;                                                 //Timer/Counter2
Overflow Inerrupt disadle
	TCCR2B = 0x00;                                                 //Timer/Counter
stopped

	message_gps = 0;
	UCSR0B |= (1<<RXCIE0);            //RX0 Complete Interrupt Enable
//	UCSR1B |= (1<<RXCIE1);            //RX1 Complete Interrupt Enable
	}

//*****************************************************************************
void ChangeFreq(void){
	if (Counter2 == COUNTER_1200)
		{
		Counter2 = TCNT2;
		TCNT2 =((((Counter2 - COUNTER_1200)>>1)) + COUNTER_2200);
		Counter2 = COUNTER_2200;                       // 2200 Hz
		}
		else
			{
			Counter2 = TCNT2;
			if (Counter2 >= 247)              //(Counter2 >= 252)
				{
				TCNT2 = COUNTER_1200;
				ChangePhase();
			//	TCNT2 = COUNTER_1200+(((((Counter2 - COUNTER_2200)<<1)) +
COUNTER_1200)-256);
				}
				else
					TCNT2 = ((((Counter2 - COUNTER_2200)<<1)) +
COUNTER_1200);
			Counter2 = COUNTER_1200;                // 1200 Hz

			}
}

//*****************************************************************************
void ax25crcBit(int lsb_int){
	static unsigned short	xor_int;

	xor_int = crc ^ lsb_int;                       // XOR lsb of CRC with the
latest bit
	crc >>= 1;                                                 // Shift 16-bit CRC
one bit to the right

	if (xor_int & 0x0001)                                  // If XOR result from
above has lsb set
		{
		crc ^= 0x8408;                                         // Shift 16-bit CRC
one bit to the right
```

```
            }
}

//*****************************************************************************
void ax25address(unsigned char txbyteA){
       ax25sendByte(txbyteA<<1);
       }

//*****************************************************************************
void m_puts(unsigned char *text2 ){
       unsigned char i=0,temp2;
       do
              {
              temp2 = text2[i];
              if(temp2==0) break;
              ax25sendByte(temp2);
              i++;
              }
       while(temp2>0);
  }

//*****************************************************************************
void ChangePhase(void){
       if (Phase == TRUE)
              {
              PORTB |= TxDxData;          //PC2 = 1
              Phase = FALSE;
              }
         else
              {
              PORTB &= ~TxDxData;    //PC2 = 0;
              Phase = TRUE;
              }
       }
//*****************************************************************************
void ax25sendCrc(void){
       static unsigned char crchi;

       crchi = (crc >> 8)^0xFF;
       ax25sendByte(crc^0xFF);                          // Send the low byte of the crc
       ax25sendByte(crchi);                                    // Send the high byte of
the crc
       ax25sendByte(0x7E);                                     // Send a flag to end the
packet

       crc = 0xFFFF;
       }
```

Soubor **Adf7021.c**

```
#include "ADF7021.h"
#include "main.h"

//*****************************************************************************
void config_use_defaults(void)
  {
       //write R0, turn on PLL
       r0_tx_H = 0x10D1;          //0x10D0
       r0_tx_L = 0x4610;          //0x5550
       //write R1, turn on VCO
       r1_tx_H = 0x0396;          //0x021F
```

```
        r1_tx_L = 0xD391;           //0x5291
        //write R2, turn on PA
        r2_tx_H = 0x0090;           //0x00C0
        r2_tx_L = 0xF0C2;           //0xF082
        //write R3, turn on TX/RX clocks
        r3_tx_H = 0x29BC;           //0x35BC
        r3_tx_L = 0x3913;           //0x74D3
  }

//*****************************************************************************
void dd_adf7020_chip_sel(char cs)
        {
   if (cs) {
              PORTC |= ADF7020_CE;    //ADF7020_CE = 1;
//            u_puts1(" \r\n CE = 1 ");
              }
        else {
                      PORTC &= ~ADF7020_CE;        //ADF7020_CE = 0;
//                    u_puts1(" \r\n CE = 0 ");
                      }
        }
//*****************************************************************************
void ADF7021_set()
        {
        dd_adf7020_chip_sel(1);     //ADF7020 CE high
        _delay_ms(1);                       //na osc. 1800us
        //write R1, turn on VCO
        dd_write_7020_reg(21);              //Write to register R1 Tx 1 + 20 = 21
        _delay_us(800);                             //na osc. 1800us
        //write R3, turn on TX/RX clocks
        dd_write_7020_reg(3);               //Write to register R3 Rx 3 + 00 = 03
        //write R0, turn on PLL
        dd_write_7020_reg(20);              //Write to register R0 Tx 0 + 20 = 20
        _delay_us(40);                              //na osc. 54us
        //write R2, turn on PA
        dd_write_7020_reg(22);              //Write to register R2 Tx 2 + 20 = 22
}

//*****************************************************************************
void dd_write_7020_reg(char ADa)
        {
    char ADb, ADc;
    unsigned int byte, byte_H, byte_L;

       switch(ADa)
              {
              case 3: {     byte_H = r3_tx_H;
                            byte_L = r3_tx_L;} break;
              case 4: {     byte_H = r4_tx_H;
                            byte_L = r4_tx_L;} break;
              case 20: {    byte_H = r0_tx_H;
                            byte_L = r0_tx_L;} break;
              case 21: {    byte_H = r1_tx_H;
                            byte_L = r1_tx_L;} break;
              case 22: {    byte_H = r2_tx_H;
                            byte_L = r2_tx_L;} break;
                     }
/*
       u_puts1("\r\n");
       if (((byte_H & 0xFF00) >> 8) <= 0x0F) u_putc1('0');
       itoa(((byte_H & 0xFF00) >> 8),tempstr,16);// číslo int1 převeď na string
tempstr,
```

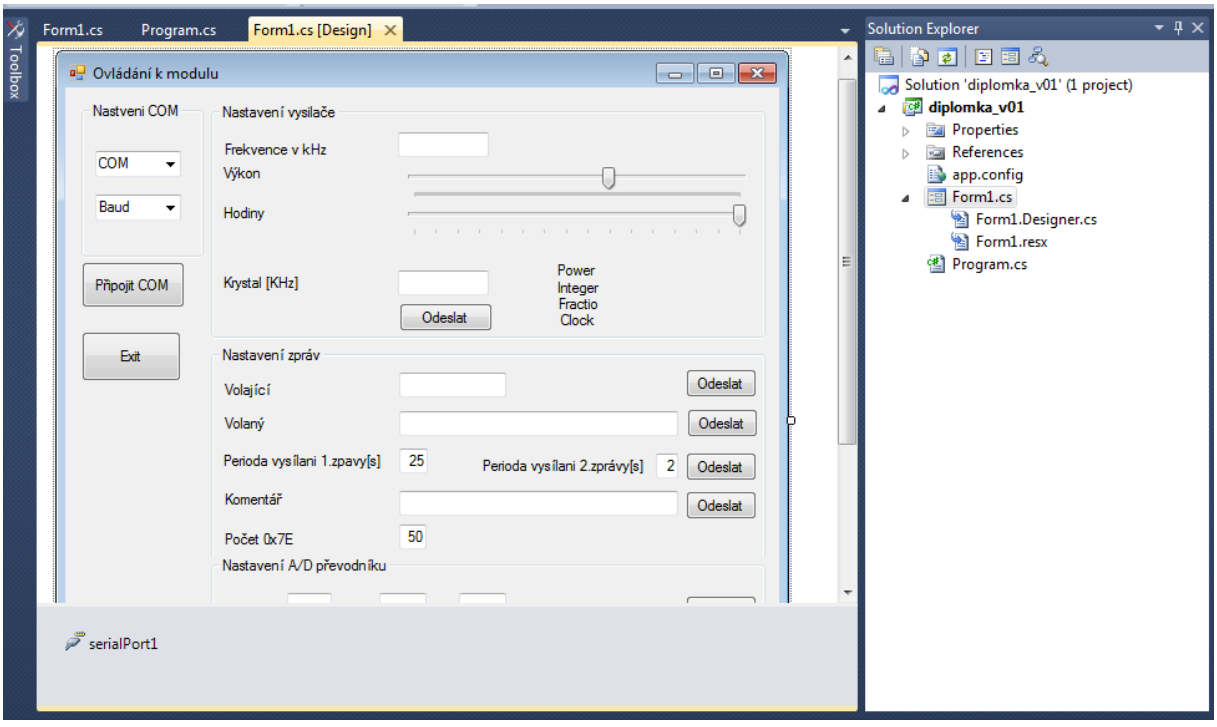
```
	u_puts1(tempstr);
	if (((byte_H & 0x00FF)) <= 0x0f) u_putc1('0');
	itoa(((byte_H & 0x00FF)),tempstr,16);     // číslo int1 převeď na string
tempstr,
	u_puts1(tempstr);
	if (((byte_L & 0xFF00) >> 8) <= 0x0F) u_putc1('0');
	itoa(((byte_L & 0xFF00) >> 8),tempstr,16);// číslo int1 převeď na string
tempstr,
	u_puts1(tempstr);
	if (((byte_L & 0x00FF)) <= 0x0f) u_putc1('0');
	itoa(((byte_L & 0x00FF)),tempstr,16);     // číslo int1 převeď na string
tempstr,
	u_puts1(tempstr);
	u_puts1("        ");
*/
	PORTC &= ~ADF7020_SLE;                                  //ADF7020_SLE = 0;
	PORTC &= ~ADF7020_SCLK;                                 //ADF7020_SCLK = 0;

	for (ADb = 0; ADb < 2; ADb++)
		{
		switch(ADb)
			{
			case 0: {byte = byte_H;} break;
			default : {byte = byte_L;} break;
			}
	    for (ADc=16; ADc > 0; ADc--)
			{
			PORTC &= ~ADF7020_SCLK;
			_delay_us(2);                                       //_delay_ms(2);
		     if (byte & 0x8000)
				{
				PORTC |= ADF7020_SDATA;                 //ADF7020_SDATA =
1; PORTC = (PORTC | 0b00000010);
				}
			else
					{
					PORTC &= ~ADF7020_SDATA;    //ADF7020_SDATA = 0;
PORTC = (PORTC & 0b11111101);
					}

			PORTC |= ADF7020_SCLK ;                 //ADF7020_SCLK = 1;
			byte = (byte<<1);                               // left shift 1
			_delay_us(2);
			}
		PORTC &= ~ADF7020_SCLK;
		}
	PORTC |= ADF7020_SLE;                                           //ADF7020_SLE = 1;
// Slight pulse extend
	_delay_us(2);
	PORTC &= ~ADF7020_SDATA;                                //ADF7020_SDATA = 0;
	PORTC &= ~ADF7020_SLE;                                          //ADF7020_SLE = 0;
	}
```

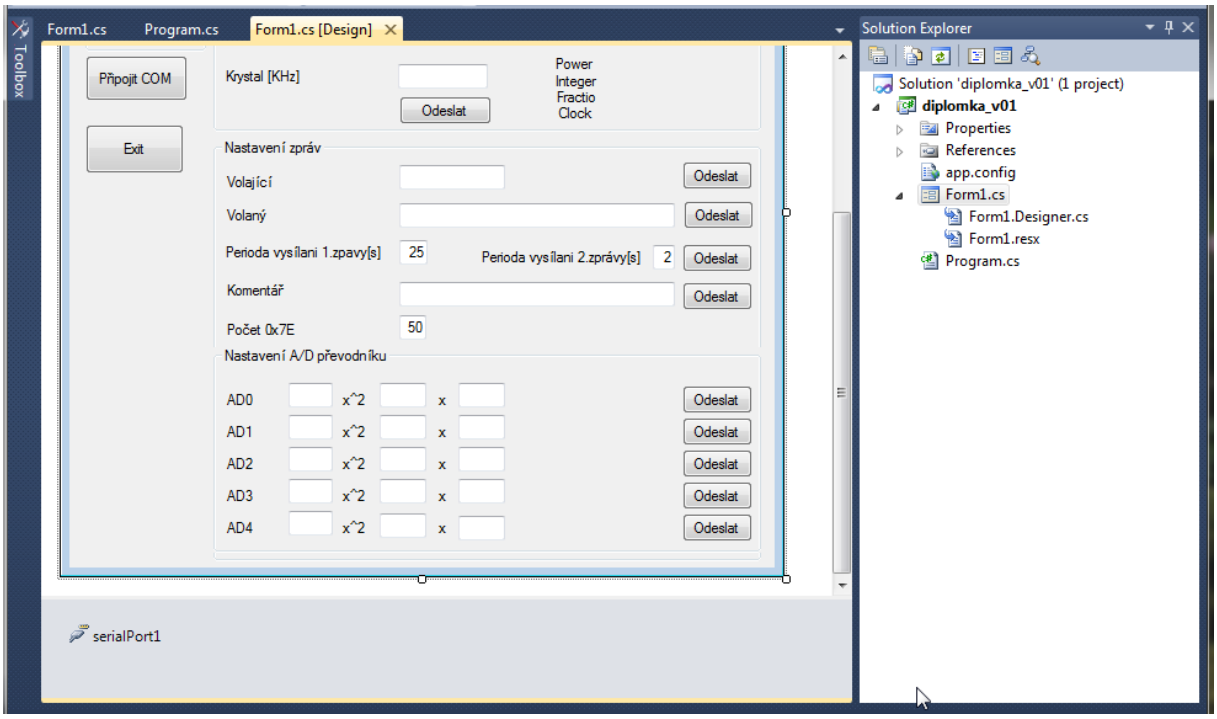
Soubor **Adf7021.h**

```
#include "main.h"

//*********************************************************************
```

```
//*                      Hardware pin mapping                                  *
//*****************************************************************************/

#define ADF7020_SDATA      0b00010000;   // PIN1 Output
#define ADF7020_SLE        0b00100000;   // PIN2 Output
#define ADF7020_CE         0b10000000;   // PIN4 Output

//#define ADF7020_SREAD      PINB,5;           // PIN3 Input
#define ADF7020_SCLK       0b00000100;   // PIN0 Output
//#define ADF7020_MUXOUT     PINB,7;           // PIN5 Input

//unsigned int r1_rx_H,r1_rx_L;

unsigned int r0_tx_H,r0_tx_L;
unsigned int r1_tx_H,r1_tx_L;
unsigned int r2_tx_H,r2_tx_L;
unsigned int r3_tx_H,r3_tx_L;
unsigned int r4_tx_H,r4_tx_L;


void dd_write_7020_reg(char a);
void dd_adf7020_chip_sel(char cs);
void dd_initialise(void);
void dd_short_delay(int count);

//void u_putc( char data );                  //napiÜ znak do UART
//void u_puts( char *text );                 //napiÜ string do UART


/* EOF */
```

A ovládací program napsaný v MS Visual Studio






```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;

namespace diplomka_v01
{

    public partial class Form1 : Form
    {
        string RxString;    // Add this variable
        int PFD = 11059;
        int Krystal = 11059;
        int Inter_N = 23;
        Int32 Fract_N = 1;
        Int32 Freq = 144000;
        int Power = 0;
        int Power1;
        int bbb;
        int BaudRate = 4800;
        int PE1 = 2;
        int PE2 = 2;
        int STA = 50;
        int AD0a, AD0b, AD0c;
        int AD1a, AD1b, AD1c;
        int AD2a, AD2b, AD2c;
        int AD3a, AD3b, AD3c;
```

```
        int AD4a, AD4b, AD4c;



        public Form1()
        {
            InitializeComponent();
            texBFrek.Text = Freq.ToString();
            trackBar2.Value = Power;
            labFreq.Text = "Frekvence v Khz";
            labPow.Text = "Výkon v dBm";
            texBKry.Text = Krystal.ToString();
            ComEnable(false);

            ViewCom();

            comBBaud.Items.Add(4800);
            comBBaud.Items.Add(9600);
            comBBaud.Items.Add(57600);
            comBBaud.Items.Add(115200);

            comBBaud.Text = BaudRate.ToString();
        }

        private void ViewCom()
        {
            comBCOM.Items.Clear();
            foreach (string s in SerialPort.GetPortNames())
            {
                comBCOM.Items.Add(s);
            }
        }

        private void trackBar2_Scroll(object sender, EventArgs e)
        {
            //  Freq = Int32.Parse(texBFrek.Text);
            Power = (trackBar2.Value / 10);
            labPow.Text = "Výkon je " + Power.ToString() + " dBm";
            Vypocet();
        }

        private void butComOpen_Click(object sender, EventArgs e)
        {
            try
            {
                if (butComOpen.Text == "Close COM")
                    Close_COM();
                else
                {
                    ViewCom();
                    Open_COM();
                }
            }
            catch
            {
                MessageBox.Show("Chyba pri pripojení COM");
                ComEnable(false);
            }
            System.Threading.Thread.Sleep(200);
        }

        private void Open_COM()
```

```
        {
            butComOpen.Text = "Odpojit COM";

            serialPort1.PortName = comBCOM.Text;

            serialPort1.BaudRate = BaudRate;
            serialPort1.Open();
            if (serialPort1.IsOpen)
            {
                ComEnable(true);
                serialPort1.Write("Hello, world");
            }
            else Close_COM();
        }


        private void Close_COM()
        {
            serialPort1.Close();
            ComEnable(false);
        }

        private void button1_Click(object sender, EventArgs e)
        {

            Fract_N = 1;
            Inter_N = 23;
            Vypocet();
            DataSend();
        }

        private void Vypocet()
        {
            Power1 = ((Power + 16) * 2);
            labePOW.Text = Power1.ToString();
            PFD = (Krystal) / 1;
            bbb = 0;
            while (bbb == 0)
            {
                float aaa = (2 * Freq);
                float ddd = (aaa / PFD);
                float ccc = (ddd - Inter_N);
                Fract_N = (int)((32768 * ccc));
                if (Fract_N > 32768) Inter_N++;
                if (Fract_N < 0) Inter_N--;
                if ((Fract_N >= 0) && (Fract_N < 32768)) bbb = 1;
                if ((Inter_N < 23) | (Fract_N < 0))
                {
                    bbb = 1;
                }
            }
            if ((Inter_N < 23) | (Fract_N < 0))
            {
                labePOW.Text = "Mimo rozsah";
                labINT.Enabled = false;
                labFRA.Enabled = false;
                butSend.Enabled = false;
            }
            else
            {
                labINT.Enabled = true;
                labFRA.Enabled = true;
```

```
                labePOW.Text = "Power " + Power1.ToString();
                labINT.Text = "Integer_N " + Inter_N.ToString();
                labFRA.Text = "Fractio_N " + Fract_N.ToString();
                butSend.Enabled = true;
            }

            if (trackBar3.Value > 0)
            {
                labClo.Text = "Clock " + ((Krystal / (2 *
trackBar3.Value))).ToString() + " KHz";
            }
            else
            {
                labClo.Text = "Clock OFF";
            }
        }

        private void DataSend()
        {
            if (serialPort1.IsOpen)
            {
                serialPort1.Write("\r\n$POW " + (Power1.ToString()) + ";");
                System.Threading.Thread.Sleep(60);
                serialPort1.Write("\r\n$INT " + (Inter_N.ToString()) + ";");
                System.Threading.Thread.Sleep(60);
                serialPort1.Write("\r\n$FRA " + (Fract_N.ToString()) + ";");
                System.Threading.Thread.Sleep(60);
                serialPort1.Write("\r\n$CLO " + (trackBar3.Value) + ";");
                System.Threading.Thread.Sleep(60);
                serialPort1.Write("\r\n$SET 1;");
            }
            else
            {
                MessageBox.Show("Neni otevreny COM");
                Close_COM();
            }
        }


        private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            RxString = serialPort1.ReadExisting();
          //  this.Invoke(new EventHandler(DisplayText));
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            serialPort1.Close();
        }

        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            serialPort1.Close();
        }


        private void comBBaud_SelectedIndexChanged(object sender, EventArgs e)
        {
            BaudRate = Int32.Parse(comBBaud.Text);
        }
```

```
        private void ComEnable(bool SetCOM)
        {
            if (SetCOM)
            {
                groBCom.Enabled = false;
                groBSet.Enabled = true;
                groBZprava.Enabled = true;
                butComOpen.Text = "Close COM";

            }
            else
            {
                groBCom.Enabled = true;
                groBSet.Enabled = false;
                groBZprava.Enabled = false;
                butComOpen.Text = "Open COM";
            }
        }

        private void butExit_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen) serialPort1.Close();
            Application.Exit();
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            try
            {
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$DES " + (texBDES.Text)
+ ";");
                System.Threading.Thread.Sleep(60);
            }
            catch
            {
                MessageBox.Show("Není otevřený port!");
                Close_COM();
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            try
            {
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$SOU " + (texBSOU.Text)
+ ";");
                System.Threading.Thread.Sleep(60);
            }
            catch
            {
                MessageBox.Show("Není otevřený port!");
                Close_COM();
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            try
            {
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$PE1 " + PE1 + ";");
                System.Threading.Thread.Sleep(60);
```

```
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$PE2 " + PE2 + ";");
                System.Threading.Thread.Sleep(60);
            }
            catch
            {
                MessageBox.Show("Není otevřený port!");
                Close_COM();
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            try
            {
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$COM " + (texBCOM.Text)
+ ";");
                System.Threading.Thread.Sleep(60);
            }
            catch
            {
                MessageBox.Show("Není otevřený port!");
                Close_COM();
            }
        }

        private void texBPE1_TextChanged(object sender, EventArgs e)
        {
            try
            {
                PE1 = Int16.Parse(texBPE1.Text);
            }
            catch
            {
                MessageBox.Show("Musí to být celé číslo PE1");
            }
        }

        private void texBPE2_TextChanged(object sender, EventArgs e)
        {
            try
            {
                PE2 = Int16.Parse(texBPE2.Text);
            }
            catch
            {
                MessageBox.Show("Musí to být celé číslo PE2");
            }
        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {
            try
            {
                STA = Int16.Parse(texBSTA.Text);
            }
            catch
            {
                MessageBox.Show("Musí to být celé číslo STA");
            }
        }

        private void texBFrek_TextChanged(object sender, EventArgs e)
```

```
        {
            try
            {
                Freq = Int32.Parse(texBFrek.Text);
                Krystal = Int16.Parse(texBKry.Text);
                Vypocet();
            }
            catch
            {
                if (serialPort1.IsOpen) MessageBox.Show("Musí to být celé číslo Vyp");
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            try
            {
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$STA " + STA + ";");
                System.Threading.Thread.Sleep(60);
            }
            catch
            {
                MessageBox.Show("Není otevřený port!");
                Close_COM();
            }
        }

        private void butAD0_Click(object sender, EventArgs e)
        {
            try
            {
                AD0a = Int16.Parse(texAD0a.Text);
                AD0b = Int16.Parse(texAD0b.Text);
                AD0c = Int16.Parse(texAD0c.Text);
            }
            catch
            {
                MessageBox.Show("Musí to být celé číslo AD0");
            }

            try
            {
                //if (serialPort1.IsOpen) serialPort1.Write("\r\n$AD0 " + AD0a + "," +
AD0b + "," + AD0c + ",;");
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$AD0 ");
                    if (AD0a >1000) serialPort1.Write(AD0a + ",");
                        else
                            {
                            serialPort1.Write("0");
                             if (AD0a >100) serialPort1.Write(AD0a + ",");
                             else
                                { serialPort1.Write("0");
                                 if (AD0a >10) serialPort1.Write(AD0a + ",");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD0a + ",");
                                     }
                                 }
                            }
                    if (AD0b >1000) serialPort1.Write(AD0b + ",");
                        else
```

```
                            {
                            serialPort1.Write("0");
                             if (AD0b >100) serialPort1.Write(AD0b + ",");
                             else
                                { serialPort1.Write("0");
                                if (AD0b > 10) serialPort1.Write(AD0b + ",");
                                    else
                                        {
                                            serialPort1.Write("0");
                                            serialPort1.Write(AD0b + ",");
                                        }
                                }
                            }
                        if (AD0c >1000) serialPort1.Write(AD0c + ",;");
                           else
                            {
                            serialPort1.Write("0");
                             if (AD0c >100) serialPort1.Write(AD0c + ",");
                             else
                                { serialPort1.Write("0");
                                 if (AD0c >10) serialPort1.Write(AD0c + ",;");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD0c + ",;");
                                     }
                                }
                            }
                     }
                System.Threading.Thread.Sleep(60);
            }
            catch
            {
                MessageBox.Show("Není otevřený port!");
                Close_COM();
            }
        }

        private void texAD4c_TextChanged(object sender, EventArgs e)
        {

        }

        private void butAD1_Click(object sender, EventArgs e)
        {
            try
            {
               AD1a = Int16.Parse(texAD1a.Text);
               AD1b = Int16.Parse(texAD1b.Text);
               AD1c = Int16.Parse(texAD1c.Text);
            }
            catch
            {
                MessageBox.Show("Musí to být celé čísloAD1");
            }

            try
            {
                //if (serialPort1.IsOpen) serialPort1.Write("\r\n$AD1 " +AD1a + ","
+AD1b + "," +AD1c + ",;");
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$AD1 ");
                    if (AD1a >1000) serialPort1.Write(AD1a + ",");
                        else
```

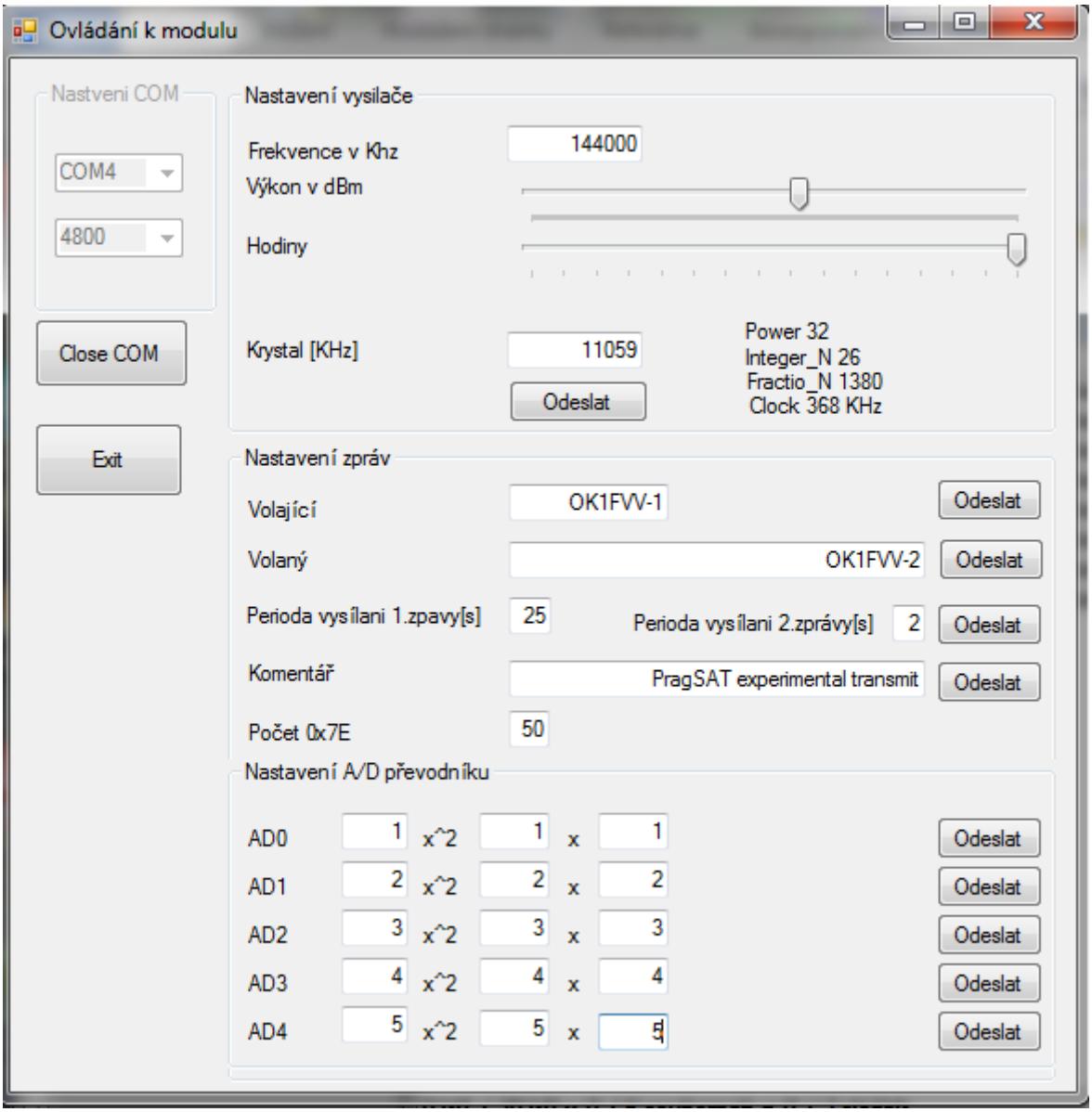
```
                        {
                        serialPort1.Write("0");
                         if (AD1a >100) serialPort1.Write(AD1a + ",");
                         else
                            { serialPort1.Write("0");
                             if (AD1a >10) serialPort1.Write(AD1a + ",");
                             else
                                 {
                                     serialPort1.Write("0");
                                     serialPort1.Write(AD1a + ",");
                                 }
                             }
                        }
                if (AD1b >1000) serialPort1.Write(AD1b + ",");
                    else
                        {
                        serialPort1.Write("0");
                         if (AD1b >100) serialPort1.Write(AD1b + ",");
                         else
                            { serialPort1.Write("0");
                             if (AD1b >10) serialPort1.Write(AD1b + ",");
                             else
                                 {
                                     serialPort1.Write("0");
                                     serialPort1.Write(AD1b + ",");
                                 }
                            }
                        }
                 if (AD1c >1000) serialPort1.Write(AD1c + ",;");
                    else
                        {
                        serialPort1.Write("0");
                         if (AD1c >100) serialPort1.Write(AD1c + ",");
                         else
                            { serialPort1.Write("0");
                             if (AD1c >10) serialPort1.Write(AD1c + ",;");
                             else
                                 {
                                     serialPort1.Write("0");
                                     serialPort1.Write(AD1c + ",;");
                                 }
                            }
                         }
            System.Threading.Thread.Sleep(60);
        }
        catch
        {
            MessageBox.Show("Není otevřený port!");
            Close_COM();
        }
    }

    private void butAD2_Click(object sender, EventArgs e)
    {
     try
        {
           AD2a = Int16.Parse(texAD2a.Text);
           AD2b = Int16.Parse(texAD2b.Text);
           AD2c = Int16.Parse(texAD2c.Text);
        }
        catch
        {
```

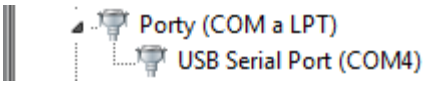
```
                MessageBox.Show("Musí to být celé čísloAD2");
            }

            try
            {
                //if (serialPort1.IsOpen) serialPort1.Write("\r\n$AD2 " +AD2a + ","
+AD2b + "," +AD2c + ",;");
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$AD2 ");
                    if (AD2a >1000) serialPort1.Write(AD2a + ",");
                        else
                            {
                            serialPort1.Write("0");
                             if (AD2a >100) serialPort1.Write(AD2a + ",");
                             else
                                { serialPort1.Write("0");
                                 if (AD2a >10) serialPort1.Write(AD2a + ",");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD2a + ",");
                                     }
                                 }
                            }
                    if (AD2b >1000) serialPort1.Write(AD2b + ",");
                        else
                            {
                            serialPort1.Write("0");
                             if (AD2b >100) serialPort1.Write(AD2b + ",");
                             else
                                { serialPort1.Write("0");
                                 if (AD2b >10) serialPort1.Write(AD2b + ",");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD2b + ",;");
                                     }
                                }
                            }
                     if (AD2c >1000) serialPort1.Write(AD2c + ",;");
                        else
                            {
                            serialPort1.Write("0");
                             if (AD2c >100) serialPort1.Write(AD2c + ",");
                             else
                                { serialPort1.Write("0");
                                 if (AD2c >10) serialPort1.Write(AD2c + ",;");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD2c + ",");
                                     }
                                }
                             }
                System.Threading.Thread.Sleep(60);
            }
            catch
            {
                MessageBox.Show("Není otevřený port!");
                Close_COM();
            }
        }
```

```
        private void butAD3_Click(object sender, EventArgs e)
        {
             try
            {
               AD3a = Int16.Parse(texAD3a.Text);
               AD3b = Int16.Parse(texAD3b.Text);
               AD3c = Int16.Parse(texAD3c.Text);
            }
            catch
            {
                MessageBox.Show("Musí to být celé čísloAD3");
            }

            try
            {
                //if (serialPort1.IsOpen) serialPort1.Write("\r\n$AD3 " +AD3a + ","
+AD3b + "," +AD3c + ",;");
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$AD3 ");
                    if (AD3a >1000) serialPort1.Write(AD3a + ",");
                        else
                            {
                            serialPort1.Write("0");
                             if (AD3a >100) serialPort1.Write(AD3a + ",");
                             else
                                { serialPort1.Write("0");
                                 if (AD3a >10) serialPort1.Write(AD3a + ",");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD3a + ",");
                                     }
                                 }
                            }
                    if (AD3b >1000) serialPort1.Write(AD3b + ",");
                        else
                            {
                            serialPort1.Write("0");
                             if (AD3b >100) serialPort1.Write(AD3b + ",");
                             else
                                { serialPort1.Write("0");
                                 if (AD3b >10) serialPort1.Write(AD3b + ",");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD3b + ",");
                                     }
                                }
                            }
                     if (AD3c >1000) serialPort1.Write(AD3c + ",;");
                        else
                            {
                            serialPort1.Write("0");
                             if (AD3c >100) serialPort1.Write(AD3c + ",");
                             else
                                { serialPort1.Write("0");
                                 if (AD3c >10) serialPort1.Write(AD3c + ",;");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD3c + ",;");
                                     }
                                }
                            }
```

```
                        }
                System.Threading.Thread.Sleep(60);
            }
            catch
            {
                MessageBox.Show("Není otevřený port!");
                Close_COM();
            }
        }

        private void butAD4_Click(object sender, EventArgs e)
        {
             try
            {
               AD4a = Int16.Parse(texAD4a.Text);
               AD4b = Int16.Parse(texAD4b.Text);
               AD4c = Int16.Parse(texAD4c.Text);
            }
            catch
            {
                MessageBox.Show("Musí to být celé čísloAD4");
            }

            try
            {
                //if (serialPort1.IsOpen) serialPort1.Write("\r\n$AD4 " +AD4a + ","
+AD4b + "," +AD4c + ",;");
                if (serialPort1.IsOpen) serialPort1.Write("\r\n$AD4 ");
                    if (AD4a >1000) serialPort1.Write(AD4a + ",");
                        else
                            {
                            serialPort1.Write("0");
                             if (AD4a >100) serialPort1.Write(AD4a + ",");
                             else
                                { serialPort1.Write("0");
                                 if (AD4a >10) serialPort1.Write(AD4a + ",");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD4a + ",");
                                     }
                                 }
                            }
                    if (AD4b >1000) serialPort1.Write(AD4b + ",");
                        else
                            {
                            serialPort1.Write("0");
                             if (AD4b >100) serialPort1.Write(AD4b + ",");
                             else
                                { serialPort1.Write("0");
                                 if (AD4b >10) serialPort1.Write(AD4b + ",");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD4b + ",");
                                     }
                                }
                            }
                     if (AD4c >1000) serialPort1.Write(AD4c + ",;");
                        else
                            {
                            serialPort1.Write("0");
```

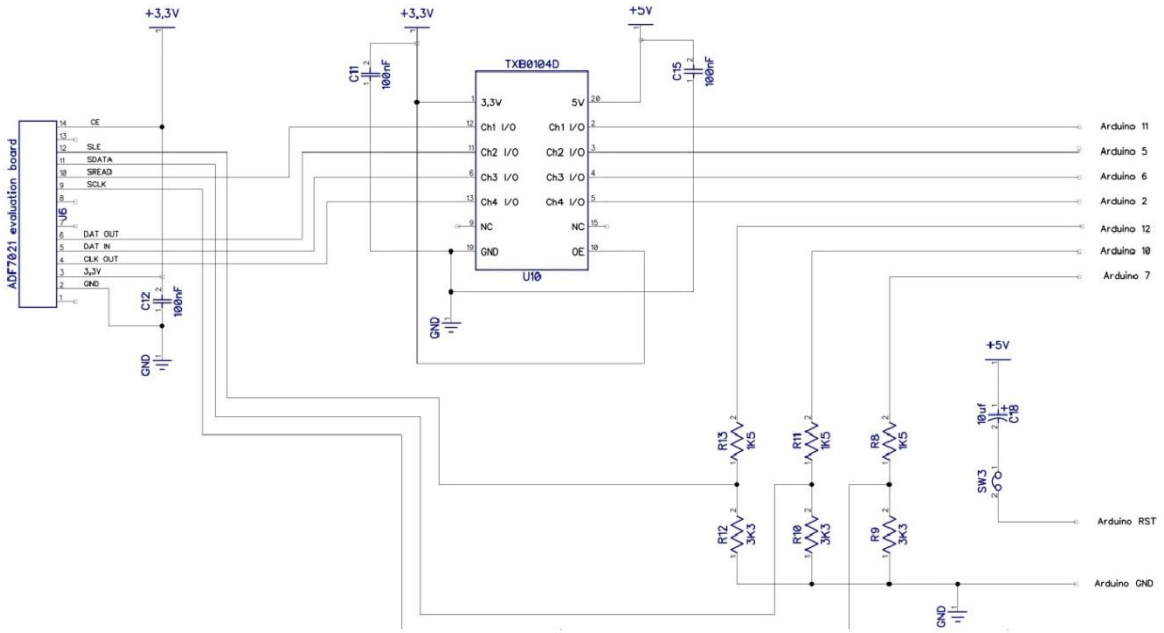
```
                            if (AD4c >100) serialPort1.Write(AD4c + ",");
                            else
                               { serialPort1.Write("0");
                                 if (AD4c >10) serialPort1.Write(AD4c + ",;");
                                 else
                                     {
                                         serialPort1.Write("0");
                                         serialPort1.Write(AD4c + ",;");
                                     }
                               }
                        }
                System.Threading.Thread.Sleep(60);
            }
            catch
            {
                MessageBox.Show("Není otevřený port!");
                Close_COM();
            }
        }
    }
}
```





4.3.2 Vysílač s Arduino

Následující zapojení má jediný účel a to převáděť napěťovou úroveň Arduina 5V na 3V3 obvodu ADF7021. Pro jednosměrný převod z V na 3V3 slouží odporové děliče. Obvod TXB0104D je použit pro oboustranný převod úrovní a jistě se dá úspěšně nahradit zapojením s tranzistory FET, které jsme použili na desce s čidly MEMS a komunikací I2C. Důvodem, proč ale toto zapojení uvádím je že je z ní zřejmé propojení signálů mezi destičkou tcvr a Arduina.



Sw pro Arduino propojené s destičkou s ADF7021 dle výše uvedeného zapojení:

```
#define pinRXCLK 2
#define pinTXCLK 3
#define pinSN 4                  // Not used
#define pinRXD 5
#define pinTXD 6
#define pinAD7021_SCLK 7
#define pinPTT 8
#define pinPLLACQ 9
#define pinAD7021_SDATA 10
#define pinAD7021_SREAD 11       // Not used
#define pinAD7021_SLE 12
#define pinLED 13

#define radio_tx_buf_len 850     // Size of radio tx ring buffer
#define radio_rx_buf_len 300     // Size of radio rx ring buffer
#define ser_in_buf_len 55        // Size of serial ring buffer
#define ser_out_buf_len 55       // Size of serial out buffer


static const uint16_t crc_table[256] PROGMEM = {

0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5,
0x60c6, 0x70e7, 0x8108, 0x9129, 0xa14a, 0xb16b,
0xc18c, 0xd1ad, 0xe1ce, 0xf1ef, 0x1231, 0x0210,
0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c,
0xf3ff, 0xe3de, 0x2462, 0x3443, 0x0420, 0x1401,
0x64e6, 0x74c7, 0x44a4, 0x5485, 0xa56a, 0xb54b,
0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6,
0x5695, 0x46b4, 0xb75b, 0xa77a, 0x9719, 0x8738,
0xf7df, 0xe7fe, 0xd79d, 0xc7bc, 0x48c4, 0x58e5,
0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969,
```

```
0xa90a, 0xb92b, 0x5af5, 0x4ad4, 0x7ab7, 0x6a96,
0x1a71, 0x0a50, 0x3a33, 0x2a12, 0xdbfd, 0xcbdc,
0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03,
0x0c60, 0x1c41, 0xedae, 0xfd8f, 0xcdec, 0xddcd,
0xad2a, 0xbd0b, 0x8d68, 0x9d49, 0x7e97, 0x6eb6,
0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a,
0x9f59, 0x8f78, 0x9188, 0x81a9, 0xb1ca, 0xa1eb,
0xd10c, 0xc12d, 0xf14e, 0xe16f, 0x1080, 0x00a1,
0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c,
0xe37f, 0xf35e, 0x02b1, 0x1290, 0x22f3, 0x32d2,
0x4235, 0x5214, 0x6277, 0x7256, 0xb5ea, 0xa5cb,
0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447,
0x5424, 0x4405, 0xa7db, 0xb7fa, 0x8799, 0x97b8,
0xe75f, 0xf77e, 0xc71d, 0xd73c, 0x26d3, 0x36f2,
0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9,
0xb98a, 0xa9ab, 0x5844, 0x4865, 0x7806, 0x6827,
0x18c0, 0x08e1, 0x3882, 0x28a3, 0xcb7d, 0xdb5c,
0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0,
0x2ab3, 0x3a92, 0xfd2e, 0xed0f, 0xdd6c, 0xcd4d,
0xbdaa, 0xad8b, 0x9de8, 0x8dc9, 0x7c26, 0x6c07,
0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba,
0x8fd9, 0x9ff8, 0x6e17, 0x7e36, 0x4e55, 0x5e74,
0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
};



  volatile byte input_rx_buffer;                                    // Temp input
buffer from CMX589, 8 bit buffer is filled by RX_589.
  volatile byte input_rx_buffer_bit_counter = 0;                    // When 8 bits are
received byte full wil be placed in radio_rx_buffer,
                                                                    // this is a
ringbuffer.
  volatile int  radio_rx_buffer_counter = radio_rx_buf_len;         // pointer for end
of buffer indication

  volatile byte  radio_tx_buffer[radio_tx_buf_len + 1];             // ringbuffer for
datastream from CMX589
  volatile int  radio_tx_buffer_in_counter = 0;                     // pointer for end
of buffer indication
  volatile int  radio_tx_buffer_out_counter = 0;
  volatile int radio_tx_buffer_bit_counter = 0;
   volatile int radio_tx_buffer_gen_counter = 0;

  volatile byte temp_shift_buffer[23];                              // Fifo buffer for
data analisys. 21 X 8 bits can be shift right.
  volatile int temp_shift_buffer_counter = radio_rx_buf_len - 50; // When 8 bits are
shifted a new byte pointed by counter will be
  volatile byte temp_shift_buffer_bit_counter = 0;                  // placed in
temp_shift_buffer[0].

  volatile byte header_table[28][3];                                // table that holds
header for descrambling an de-interleave
  volatile byte  header_table_counter_y;                            // x and y keep
track of actual possition.
```

```
  volatile byte  header_table_counter_x;
  volatile int  header_table_bit_counter;

  volatile byte descramble_poly;                                    // Holds initial
value for descrambling and convolutional decoding.

  volatile byte decoded_header[42];                                 // Table that holds
decoded header.
  volatile byte decoded_header_byte;                                // Temp value used
in procedure
  volatile int  decoded_header_counter;

  volatile byte serial_buffer[ser_in_buf_len + 1];                  // Input buffer for
serial communication
  volatile int serial_buffer_input_pointer = 0;
  volatile int serial_buffer_output_pointer = 0;

  volatile byte serial_out[ser_out_buf_len + 1];                    // Output buffer
for serial coomunication
  volatile int table_counter;
  volatile int serial_length;

  volatile byte frame_counter;                                      // Number of frames
to detect 21st frame
  volatile byte frame_counter_nr;

  volatile byte packet_counter;                                     // Packet counter
to stay synchronized with host
  volatile byte transmission_counter = 1;

  volatile byte rx_status = 0;                                      // Status of
transmission
  volatile byte tx_status = 0;
  volatile boolean rx_int_flag = false;
  volatile byte stream_status_modem = 0;
  volatile byte stream_status_host = 0;
  volatile boolean tx_int_flag = false;

  volatile boolean rx_inverse = false;                              // RX TX inverse
depend on type of tranceiver
  volatile boolean tx_inverse = false;

  volatile byte config_flags;                                       // Tranceiver
parameters
  volatile byte config_modulation;
  volatile byte config_TX_delay_1;
  volatile byte config_TX_delay_2;
  volatile unsigned int TX_delay_counter;
  volatile byte config_status;

  volatile byte status_flags_hi = B00000000;                        // Stream
parameters
  volatile byte status_flags_lo = B00000000;
  volatile byte status_rcv_buffer = 0;
  volatile byte status_tx_buffer = 0;
  volatile byte status_unsend_frames = 0;
  volatile byte status_total_packets_send = 0;
  volatile int status_tx_bit_counter;
  volatile byte header_flag_1;
  volatile byte header_flag_2;
  volatile byte header_flag_3;
  volatile byte preamble_counter;
```

```
  unsigned int crc;
  volatile unsigned int time_out_timer;

  volatile long AD7021_control_byte;                              // AD7021 related
  volatile int AD7021_counter;
  volatile boolean  modem_type;                                   // true = CMX589 -
false = ADF7021
  volatile byte tx_int_pin;

int freeRam () {                          // Count free ram, thanks to KI6ZUM
  extern int __heap_start, *__brkval;
  int v;
  return (int) &v - (__brkval == 0 ? (int) &__heap_start : (int) __brkval);
}



void setup(void)
{

// set all the pin modes and initial states
  pinMode(pinLED, OUTPUT);
//  pinMode(pinSN, INPUT);          // Not used
  pinMode(pinRXD, INPUT);
  pinMode(pinRXCLK, INPUT);
  pinMode(pinTXCLK, INPUT);
  pinMode(pinTXD, OUTPUT);
  pinMode(pinAD7021_SCLK, OUTPUT);
  pinMode(pinPTT, OUTPUT);
  pinMode(pinPLLACQ, OUTPUT);
//  pinMode(pinAD7021_SREAD, INPUT);        // Not used
  pinMode(pinAD7021_SLE, OUTPUT);
  pinMode(pinAD7021_SDATA, OUTPUT);

  digitalWrite(pinPTT, LOW);
  digitalWrite(pinPLLACQ, HIGH);

  digitalWrite(pinAD7021_SCLK, LOW);
  digitalWrite(pinAD7021_SLE, LOW);
  digitalWrite(pinAD7021_SDATA, LOW);


  Serial.begin(115200);                  // set serial interface to host
  Serial.flush();

#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
  Serial1.begin(115200);
  Serial1.flush();

#endif

    attachInterrupt(0, RX_589, RISING);  // Attach RX interrupt

    ADF7021_ini();                        // Initiate ADF7021

}

//==================================== Main loop
=====================================================

void loop()
```

```
{
    //------------------------------------ Serial input ------------------------------
-------------------
    // If serial char received, store in serial_buffer[x] and rotate buffer 1 pos acw
    // If valid message found Answer host and execute action, if wrong start char
delete buffer
    // If buffer pointer reach max start at min, typical ringbuffer
    // For Linux OS Please change SERIAL_BUFFER_SIZE from 64 to 128 in
HardwareSerial.cpp
    // File is located in \Arduino 1.0.X\arduino-1.0.X-windows\arduino-
1.0.X\hardware\arduino\cores\arduino

    if (Serial.available() > 0)
    {
         //  Serial1.write(Serial.available());
      serial_buffer[serial_buffer_input_pointer] = Serial.read();


      serial_buffer_input_pointer +=1;

      if (serial_buffer_input_pointer > ser_in_buf_len) serial_buffer_input_pointer =
ser_in_buf_len;
    }

    if (serial_buffer[0] != 0xD0) serial_buffer_input_pointer = 0;
    if ((serial_buffer[0] == 0xD0) && (serial_buffer_input_pointer > 4))
    {
      if (serial_buffer_input_pointer > (serial_buffer[1] + 4))
      {

        Answer_Host();
      }
    }
    //------------------------------------------------------------------------------
--------------------




  if(tx_int_flag == true)      // Calculate unsend frames
  {
      if(radio_tx_buffer_in_counter > radio_tx_buffer_out_counter)
      {
        status_unsend_frames = ((radio_tx_buffer_in_counter -
radio_tx_buffer_out_counter)/12);
      }

      if(radio_tx_buffer_in_counter < radio_tx_buffer_out_counter)
      {
        status_unsend_frames = (((radio_tx_buffer_in_counter + radio_tx_buf_len) -
radio_tx_buffer_out_counter)/12);
      }

       if(radio_tx_buffer_in_counter == radio_tx_buffer_out_counter)
      {
        status_unsend_frames = 0;
      }
  }
```

```
  if(stream_status_modem > 0)
  {

    if(stream_status_modem == 1)
    {
      if(stream_status_host > 0)
      {
        temp_shift_buffer_counter = radio_rx_buf_len - 50;
        radio_rx_buffer_counter = radio_rx_buf_len;

        detachInterrupt(0);                                 // Disable RX interrupt
        bitClear(status_flags_hi,0);                        // RX off
        bitSet(status_flags_hi,1);                          // TX on
        digitalWrite(pinPTT, HIGH);                         // Set TX on
        if(modem_type == false) ADF7021_tx();
        attachInterrupt(tx_int_pin, TX_589, FALLING);     // Enable TX interrupt
        TX_delay_counter = config_TX_delay_1 * 5;
        TX_delay_counter = TX_delay_counter + (config_TX_delay_2 * 5 * 255);
        stream_status_modem = 2;                            // Start TX delay
        tx_status = 2;

      }
    }

    if(stream_status_modem == 2)
    {
      if (TX_delay_counter < 5 ) stream_status_modem = 3;
    }

    if((stream_status_modem == 3)  && (stream_status_host > 1))
    {
      preamble_counter = 64;
      stream_status_modem = 4;
      tx_status = 3;
     // digitalWrite(pinLED, LOW);
    }

    if(stream_status_modem == 4)
    {
      if(preamble_counter < 2) stream_status_modem = 5;
    }

    if(stream_status_modem == 6)
    {
      stream_status_modem = 7;
      tx_status = 5;
    }

    if(stream_status_modem == 7)
    {
      if(time_out_timer > 4800) Set_RX();

      if(status_unsend_frames < 5)
      {
        radio_tx_buffer_out_counter -= 60;
        if(radio_tx_buffer_out_counter < 1) radio_tx_buffer_out_counter =
radio_tx_buf_len + radio_tx_buffer_out_counter;
        //radio_tx_buffer_in_counter = 240;
        //radio_tx_buffer_out_counter = 1;
      }
```

```
      if(status_unsend_frames > 66)
      {
        radio_tx_buffer_in_counter -= 60;
        if(radio_tx_buffer_in_counter < 1) radio_tx_buffer_in_counter =
radio_tx_buf_len + radio_tx_buffer_in_counter;
        //radio_tx_buffer_in_counter = 1;
        //radio_tx_buffer_out_counter = 240;
      }
    }

    if((stream_status_modem == 7) && (stream_status_host == 4))
    {
      stream_status_modem = 8;
      tx_status = 7;
    }

    if(stream_status_modem == 9) Set_RX();
  }




  if (rx_status < 6)
  {
         if (radio_rx_buffer_counter > temp_shift_buffer_counter)
         {
           if((temp_shift_buffer_counter + radio_rx_buf_len) - radio_rx_buffer_counter
> (radio_rx_buf_len-50)) rx_int_flag = true ;
         }

         if (temp_shift_buffer_counter > radio_rx_buffer_counter)
         {
           if (temp_shift_buffer_counter - radio_rx_buffer_counter >
(radio_rx_buf_len-50)) rx_int_flag = true ;
         }
  }


  if (rx_int_flag == true)
  {
    RX_buffer_shift();                                  // Rotate RX buffer

    // Wait for preamble
    if ((rx_status == 0) && (rx_int_flag == true)) Wait_for_preamble();

    // Wait for frame sync
    if ((rx_status == 1) && (rx_int_flag == true)) Wait_for_frame_sync();

    //Read radio header, descramble and de-interleav input
    if ((rx_status == 2) && (rx_int_flag == true))
    {
      Read_radio_header();
      stream_status_modem = 0;
    }

    // Convolutional decoding
    if ((rx_status == 3) && (rx_int_flag == true)) Convolutional_decode();

    // Send decoded header to derial port
    if ((rx_status == 4) && (rx_int_flag == true))
    {
```

```
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
    for (decoded_header_counter = 0; decoded_header_counter <= 40;
decoded_header_counter++)
    {
      Serial1.write(decoded_header[decoded_header_counter]);
    }

      Serial1.write(13);
      Serial1.write(10);
      Serial1.print("Free Ram : ");
      Serial1.println(freeRam(),DEC);
      Serial1.write(13);
      Serial1.write(10);
#endif
      rx_status = 5;
    }

    // Shift frame sync out of buffer to clear space
    if ((rx_status == 5) && (rx_int_flag == true))
    {
      for (frame_counter = 1; frame_counter <= 16; frame_counter +=1)
RX_buffer_shift();
      rx_status = 6;
      frame_counter = 0;
      frame_counter_nr = 21;
      rx_int_flag = false;
    }


    // RX is up and running
    if (rx_status == 6 && rx_int_flag == true)
    {
      frame_counter += 1;

       // Buffer correction, level control
       if ((frame_counter > 1) && (frame_counter < 25))     // 25 was 15
       {
         if (temp_shift_buffer_counter > radio_rx_buffer_counter)
         {
           if (temp_shift_buffer_counter - radio_rx_buffer_counter > 20 )
           {
             for (int corr = 1; corr <= 32; corr++)   // 25 was 16
             {
               RX_buffer_shift();
               frame_counter += 1;
             }
           }
         }


    if (temp_shift_buffer_counter < radio_rx_buffer_counter)
         {
           if ((temp_shift_buffer_counter + radio_rx_buf_len) -
radio_rx_buffer_counter > 20 )
           {
             for (int corr = 1; corr <= 32; corr++)   // 25 was 16
             {
               RX_buffer_shift();
               frame_counter += 1;
             }
           }
         }
```

```
       }


      // Check termination flags
      if (frame_counter ==73)
      {
        if (temp_shift_buffer[12] == B01010101
         && temp_shift_buffer[11] == B01010101
         && temp_shift_buffer[10] == B01010101
          && temp_shift_buffer[9] == B01010101
          && temp_shift_buffer[8] == B11001000
          && temp_shift_buffer[7] == B01111010)
        {
          Set_RX();
          Send_EOT();
 //         #if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
 //           Serial1.print("EOT found");
 //           Serial1.write(13);
 //           Serial1.write(10);
//#endif
         }
      }

      // Check on gained bit
      if (frame_counter_nr == 20 && frame_counter >=66)
      {
        if (temp_shift_buffer[11] == B01010101
         && temp_shift_buffer[10] == B00101101
          && temp_shift_buffer[9] == B00010110)
        {
//#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
 //         Serial1.print("shift");
 //         Serial1.write(13);
 //         Serial1.write(10);
//#endif
          rx_int_flag = false;
          RX_buffer_shift();
        }
      }

      // Check sync flag
      if (frame_counter_nr == 20 && frame_counter ==73)
        {
          if (temp_shift_buffer[12] == B01010101
           && temp_shift_buffer[11] == B00101101
           && temp_shift_buffer[10] == B00010110)
          {
          }
          // If 1 out of 3 is okay, accept and correct
          else if (temp_shift_buffer[12] == B01010101
              || temp_shift_buffer[11] == B00101101
              || temp_shift_buffer[10] == B00010110)
          {
            temp_shift_buffer[12] = B01010101;
            temp_shift_buffer[11] = B00101101;
            temp_shift_buffer[10] = B00010110;
//#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
//            Serial1.print("sync gecorrigeerd");
//            Serial1.write(10);
//#endif
          }
```

```
          // Resync
          else
          {
            Set_RX();
            Send_EOT();
 //           #if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
 //           Serial1.print("Geen sync gevonden");
 //           Serial1.write(13);
 //           Serial1.write(10);
//#endif

          }
        }

        // Send voice / data Packet to host
        if (frame_counter ==73)
        {
          packet_counter = frame_counter_nr;
          Send_Packet_to_Host();

        }

        // Count frames
        if (frame_counter >= 96)
        {
          frame_counter = 0;
          frame_counter_nr += 1;
          if (frame_counter_nr > 20)
          {
            frame_counter_nr = 0;
          }
        }
      }


    } // end if rx_int_flag == true



  rx_int_flag = false;
  tx_int_flag = false;


} // end main loop

//========================================== Send AD7021 byte
=====================================================
void Send_AD7021_control()
{
  for(AD7021_counter = 31; AD7021_counter >= 0; AD7021_counter--)
  {
    if(bitRead(AD7021_control_byte, AD7021_counter) == HIGH)
    {
      digitalWrite(pinAD7021_SDATA, HIGH);
    }
    else
    {
      digitalWrite(pinAD7021_SDATA, LOW);
    }

    digitalWrite(pinAD7021_SCLK, HIGH);
```

```
//    delayMicroseconds(5);
    digitalWrite(pinAD7021_SCLK, LOW);
  }

  digitalWrite(pinAD7021_SLE, HIGH);
  digitalWrite(pinAD7021_SLE, LOW);
  digitalWrite(pinAD7021_SDATA, LOW);

}

//======================================== Set RX
============================================================
void Set_RX()
{
      detachInterrupt(tx_int_pin);             // Disable TX interrupt
      attachInterrupt(0, RX_589, RISING);      // Enable RX interrupt
      bitClear(status_flags_hi,0);             // RX on
      bitClear(status_flags_hi,1);             // TX off
      digitalWrite(pinPTT, LOW);               // Set TX off

      if(modem_type == false) ADF7021_rx();

      radio_rx_buffer_counter = radio_rx_buf_len;
 //     temp_shift_buffer_counter = 75;
 //     input_rx_buffer_bit_counter = 0;
      temp_shift_buffer_bit_counter = 0;
      rx_status = 0;
      digitalWrite(pinPLLACQ, HIGH);
      frame_counter = 0;
      frame_counter_nr = 0;
      status_total_packets_send = 0;
      status_unsend_frames = 0;

      stream_status_modem = 1;
      stream_status_host = 0;
      tx_status = 1;
}
//======================================== Send EOT
============================================================
void Send_EOT()
{
    serial_out[2] = 0x03;
    serial_out[3] = 0x00;
    serial_out[4] = 0x1A;
    serial_out[5] = transmission_counter;
    serial_out[6] = packet_counter;

    serial_length = 6;
    Send_Serial();
}
//======================================== Send packet to host
================================================
void Send_Packet_to_Host()
{
    serial_out[2] = 0x13;
    serial_out[3] = 0x00;
    serial_out[4] = 0x19;
    serial_out[5] = transmission_counter;
    serial_out[6] = packet_counter;
    serial_out[7] = 0xE8;
    serial_out[8] = 0x03;
       for (int d = 1; d <= 12; d++)
```

```
        {
          serial_out[8 + d] = temp_shift_buffer[22 - d];
        }
    serial_out[21] = 0x00;
    serial_out[22] = 0x00;

#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
 //            Serial1.print(radio_rx_buffer_counter);
 //            Serial1.print("-");
 //            Serial1.print(temp_shift_buffer_counter);
 //            Serial1.write(13);
 //            Serial1.write(10);
#endif


    serial_length = 22;
    Send_Serial();

}
//======================================== List of answers to host
==========================================
void Answer_Host()
{
  if (serial_buffer[3] == 0x11)                       // Answer to get version
  {
    serial_out[2] = 0x11;
    serial_out[3] = 0x00;
    serial_out[4] = 0x91;
    serial_out[5] = 0x22;          //2B
    serial_out[6] = 0x11;          //1.1
    serial_out[7] = 'T';
    serial_out[8] = 'Y';
    serial_out[9] = 'M';
    serial_out[10] = ' ';
    serial_out[11] = 'P';
    serial_out[12] = 'r';
    serial_out[13] = 'g';
    serial_out[14] = 'S';
    serial_out[15] = 'A';
    serial_out[16] = 'T';
    serial_out[17] = '2';
    serial_out[18] = '0';
    serial_out[19] = '1';
    serial_out[20] = '4';

    serial_length = 20;
    Send_Serial();


     // Serial.print("Free Ram : ");
     // Serial.println(freeRam(),DEC);
     // Serial.write(13);
     // Serial.write(10);


    stream_status_modem = 1;   // modem TX is idle
    tx_status = 1;
//#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
//            Serial1.print("Get version");
 //           Serial1.write(13);
 //           Serial1.write(10);
//#endif
```

```
  }


  if (serial_buffer[3] == 0x12)                       // Answer to get serial
  {
    serial_out[2] = 0x05;
    serial_out[3] = 0x00;
    serial_out[4] = 0x92;
    serial_out[5] = 0x13;
    serial_out[6] = 0x09;
    serial_out[7] = 0x10;
    serial_out[8] = 0x01;
    serial_length = 8;
    Send_Serial();
//#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
//            Serial1.print("Get serial");
 //           Serial1.write(13);
 //           Serial1.write(10);
//#endif
  }


  if (serial_buffer[3] == 0x14)                     // Answer to set config -> ACK
  {
   if (serial_buffer[4] == 0xC0)
   {
    config_flags = serial_buffer[6];
    config_modulation = serial_buffer[7];
    config_TX_delay_1 = serial_buffer[8];
    config_TX_delay_2 = serial_buffer[9];
    if(bitRead(config_flags,0) == HIGH) rx_inverse = true; else rx_inverse = false;
    if(bitRead(config_flags,1) == HIGH) tx_inverse = true; else tx_inverse = false;
    if(bitRead(config_flags,2) == HIGH) modem_type = false; else modem_type = true;
    if(bitRead(config_flags,2) == HIGH) tx_int_pin = 0; else tx_int_pin = 1;
    modem_type = true;
    tx_int_pin = 1;


   }
   serial_out[2] = 0x02;
   serial_out[3] = 0x00;
   serial_out[4] = 0x94;
   serial_out[5] = 0x06;
   serial_length = 5;
   Send_Serial();
  }

  if ((serial_buffer[3] == 0x10) && (serial_buffer[1] == 0x02))  // Answer to set
status -> ACK
  {
    config_status = serial_buffer[4];
    serial_out[2] = 0x02;
    serial_out[3] = 0x00;
    serial_out[4] = 0x90;
    serial_out[5] = 0x06;

    serial_length = 5;
    Send_Serial();
//#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
//            Serial1.print("Answer set status");
//           Serial1.write(10);
//#endif
```

```
  }

  if ((serial_buffer[3] == 0x10) && (serial_buffer[1] == 0x01))  // Answer to get
status -> status flags
  {


//#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
//
//             Serial1.print("*");
//             Serial1.print(status_unsend_frames);
//             Serial1.write(13);
//             Serial1.write(10);
//             Serial1.print(rx_status);
//             Serial1.print("-");
//             Serial1.print(temp_shift_buffer_counter);
//             Serial1.write(13);
//             Serial1.write(10);
//#endif

 #if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
//      Serial1.write(status_unsend_frames);
      Serial1.write("#");
    Serial1.print(radio_rx_buffer_counter);
       Serial1.write("/");
     Serial1.print(temp_shift_buffer_counter);
 //       Serial1.write(Serial.available());
        Serial1.write("*");



 #endif


    serial_out[2] = 0x08;
    serial_out[3] = 0x00;
    serial_out[4] = 0x90;
    serial_out[5] = 0x47; // status_flags_lo
    serial_out[6] = status_flags_hi;
    serial_out[7] = tx_status;
    serial_out[8] = 0x15;
    serial_out[9] = 0xFC;
    serial_out[10] = status_unsend_frames;
    serial_out[11] = status_total_packets_send;


    serial_length = 11;
    Send_Serial();
  }


    // Start preamble
    if (serial_buffer[3] == 0x16)
  {
    if(tx_inverse == true) digitalWrite(pinTXD, HIGH); else digitalWrite(pinTXD, LOW);
    stream_status_host = 1;               // Send Preamble
  }



    // Read header from host
    if (serial_buffer[3] == 0x17)
```

```
  {

    transmission_counter = serial_buffer[4];
    packet_counter = serial_buffer[5];
//    header_flag_1 = serial_buffer[8];
//    header_flag_2 = serial_buffer[9];
//    header_flag_3 = serial_buffer[10];

    for(int rvp = 1; rvp <= 41; rvp++)      // Read uncoded header from host
    {
      decoded_header[rvp] = serial_buffer[rvp + 7];
    }

    radio_tx_buffer_in_counter = 86;
    radio_tx_buffer_out_counter = 1;


    Convolutional_encode();


    Scramble_header();
    stream_status_host = 2;

  }


    // Read voive packet from host
    if (serial_buffer[3] == 0x19)
    {
      time_out_timer = 0;
      frame_counter_nr += 1;
      if(frame_counter_nr > 20) frame_counter_nr = 0;

      for(int rvp = 1; rvp <= 12; rvp++)
      {
  //         Serial1.write(serial_buffer[rvp + 7]);
        radio_tx_buffer[radio_tx_buffer_in_counter] = serial_buffer[rvp + 7];
        radio_tx_buffer_in_counter += 1;
        if(radio_tx_buffer_in_counter > radio_tx_buf_len) radio_tx_buffer_in_counter =
1;
      }
      if(stream_status_host == 2) stream_status_host = 3;


    }




    if (serial_buffer[3] == 0x1A)
    {
      radio_tx_buffer_in_counter -= 1;
      if(radio_tx_buffer_in_counter == 0) radio_tx_buffer_in_counter =
radio_tx_buf_len;
      radio_tx_buffer_in_counter -= 1;
      if(radio_tx_buffer_in_counter == 0) radio_tx_buffer_in_counter =
radio_tx_buf_len;
      radio_tx_buffer_in_counter -= 1;
      if(radio_tx_buffer_in_counter == 0) radio_tx_buffer_in_counter =
radio_tx_buf_len;

      radio_tx_buffer[radio_tx_buffer_in_counter] = B01010101;
```

```
      radio_tx_buffer_in_counter += 1;
      if(radio_tx_buffer_in_counter > radio_tx_buf_len) radio_tx_buffer_in_counter =
1;
      radio_tx_buffer[radio_tx_buffer_in_counter] = B01010101;
      radio_tx_buffer_in_counter += 1;
      if(radio_tx_buffer_in_counter > radio_tx_buf_len) radio_tx_buffer_in_counter =
1;
      radio_tx_buffer[radio_tx_buffer_in_counter] = B01010101;
      radio_tx_buffer_in_counter += 1;
      if(radio_tx_buffer_in_counter > radio_tx_buf_len) radio_tx_buffer_in_counter =
1;
      radio_tx_buffer[radio_tx_buffer_in_counter] = B01010101;
      radio_tx_buffer_in_counter += 1;
      if(radio_tx_buffer_in_counter > radio_tx_buf_len) radio_tx_buffer_in_counter =
1;

      radio_tx_buffer[radio_tx_buffer_in_counter] = B11001000;
      radio_tx_buffer_in_counter += 1;
      if(radio_tx_buffer_in_counter > radio_tx_buf_len) radio_tx_buffer_in_counter =
1;
      radio_tx_buffer[radio_tx_buffer_in_counter] = B01111010;
      radio_tx_buffer_in_counter += 1;
      if(radio_tx_buffer_in_counter > radio_tx_buf_len) radio_tx_buffer_in_counter =
1;

      radio_tx_buffer[radio_tx_buffer_in_counter] = B00000000;
      radio_tx_buffer_in_counter += 1;
      if(radio_tx_buffer_in_counter > radio_tx_buf_len) radio_tx_buffer_in_counter =
1;
      radio_tx_buffer[radio_tx_buffer_in_counter] = B00000000;
      radio_tx_buffer_in_counter += 1;
      if(radio_tx_buffer_in_counter > radio_tx_buf_len) radio_tx_buffer_in_counter =
1;
      radio_tx_buffer[radio_tx_buffer_in_counter] = B00000000;
      radio_tx_buffer_in_counter += 1;
      if(radio_tx_buffer_in_counter > radio_tx_buf_len) radio_tx_buffer_in_counter =
1;

      stream_status_host = 4;                        // Send stop word
    }


    if (serial_buffer[3] == 0x1F)
    {
      if (serial_buffer[4] == 0xC0)
      {
        serial_out[2] = 0x06;
        serial_out[3] = 0x00;
        serial_out[4] = 0x9F;
        serial_out[5] = 0xC0;
        serial_out[6] = 0x00;
        serial_out[7] = 0x00;
        serial_out[8] = 0x00;
        serial_out[9] = 0x00;
        serial_length = 9;
        Send_Serial();
      }

      if (serial_buffer[4] == 0xC1)
      {
        serial_out[2] = 0x06;
        serial_out[3] = 0x00;
```

```
        serial_out[4] = 0x9F;
        serial_out[5] = 0xC1;
        serial_out[6] = 0x00;
        serial_out[7] = 0x00;
        serial_out[8] = 0x00;
        serial_out[9] = 0x00;
        serial_length = 9;
        Send_Serial();
      }


    }




  serial_buffer[0] = 0x00;
  serial_buffer[1] = 0x00;
  serial_buffer_input_pointer = 0;
}
//======================================== Send message to host
================================================
void Send_Serial()
{
    serial_out[1] = 0xD0;
    CALC_CRC();
    for (int i = 1; i <= serial_length; i++)
   {
     Serial.write (serial_out[i]);
   }
}
//======================================== Calculate CRC
=========================================================
void CALC_CRC()
{
    crc = 0x0000;
    for (int i = 1; i <= serial_length; i++)
    {
      crc = (crc << 8 ) ^ (pgm_read_word(crc_table + ((crc >> 8) ^ serial_out[i])));
    }
   serial_out[serial_length + 2] = (crc & 0xFF);
   serial_out[serial_length + 1] = ((crc >> 8) & 0xFF);
   serial_length += 2;
}
//======================================== Scramble header
=========================================================
void Scramble_header()
{
    header_table_counter_x= 0;
    header_table_counter_y = 0;
    descramble_poly = B11111111;
    header_table_bit_counter = 0;

    radio_tx_buffer[1] = B11101010;
    radio_tx_buffer[2] = B10100110;
    radio_tx_buffer[3] = B00000000;

    radio_tx_buffer_bit_counter = 4;
    radio_tx_buffer_gen_counter = 3;
```

```
    byte bl;

    for(bl = 0; bl <= 7; bl++)
    {
      for(header_table_bit_counter = 0; header_table_bit_counter <= 27;
header_table_bit_counter++)
      {
        if(bitRead(header_table[header_table_bit_counter][0], bl) == HIGH)
        {

bitSet(radio_tx_buffer[radio_tx_buffer_gen_counter],radio_tx_buffer_bit_counter);
        }
        else
        {

bitClear(radio_tx_buffer[radio_tx_buffer_gen_counter],radio_tx_buffer_bit_counter);
        }

        radio_tx_buffer_bit_counter += 1;
        if(radio_tx_buffer_bit_counter > 7)
        {
          radio_tx_buffer_bit_counter = 0;
          radio_tx_buffer_gen_counter += 1;
        }
      }
    }

    for(bl = 0; bl <= 3; bl++)
    {
      for(header_table_bit_counter = 0; header_table_bit_counter <= 27;
header_table_bit_counter++)
      {
        if(bitRead(header_table[header_table_bit_counter][1], bl) == HIGH)
        {

bitSet(radio_tx_buffer[radio_tx_buffer_gen_counter],radio_tx_buffer_bit_counter);
        }
        else
        {

bitClear(radio_tx_buffer[radio_tx_buffer_gen_counter],radio_tx_buffer_bit_counter);
        }

        radio_tx_buffer_bit_counter += 1;
        if(radio_tx_buffer_bit_counter > 7)
        {
          radio_tx_buffer_bit_counter = 0;
          radio_tx_buffer_gen_counter += 1;
        }
      }
    }

    for(bl = 4; bl <= 7; bl++)
    {
      for(header_table_bit_counter = 0; header_table_bit_counter <= 26;
header_table_bit_counter++)
      {
        if(bitRead(header_table[header_table_bit_counter][1], bl) == HIGH)
        {

bitSet(radio_tx_buffer[radio_tx_buffer_gen_counter],radio_tx_buffer_bit_counter);
        }
```

```
        else
        {

bitClear(radio_tx_buffer[radio_tx_buffer_gen_counter],radio_tx_buffer_bit_counter);
        }

        radio_tx_buffer_bit_counter += 1;
        if(radio_tx_buffer_bit_counter > 7)
        {
          radio_tx_buffer_bit_counter = 0;
          radio_tx_buffer_gen_counter += 1;
        }
      }
    }

    for(bl = 0; bl <= 7; bl++)
    {
      for(header_table_bit_counter = 0; header_table_bit_counter <= 26;
header_table_bit_counter++)
      {
        if(bitRead(header_table[header_table_bit_counter][2], bl) == HIGH)
        {

bitSet(radio_tx_buffer[radio_tx_buffer_gen_counter],radio_tx_buffer_bit_counter);
        }
        else
        {

bitClear(radio_tx_buffer[radio_tx_buffer_gen_counter],radio_tx_buffer_bit_counter);
        }

        radio_tx_buffer_bit_counter += 1;
        if(radio_tx_buffer_bit_counter > 7)
        {
          radio_tx_buffer_bit_counter = 0;
          radio_tx_buffer_gen_counter += 1;
        }
      }
    }

    radio_tx_buffer_bit_counter = 4;
    radio_tx_buffer_gen_counter = 3;
    for(header_table_bit_counter = 1; header_table_bit_counter <= 660;
header_table_bit_counter++)
    {
       descramble_poly = descramble_poly << 1;
       bitWrite(descramble_poly,0,((bitRead(descramble_poly, 4)) ^
(bitRead(descramble_poly, 7))));

bitWrite(radio_tx_buffer[radio_tx_buffer_gen_counter],radio_tx_buffer_bit_counter,((bi
tRead(descramble_poly, 0)) ^
(bitRead(radio_tx_buffer[radio_tx_buffer_gen_counter],radio_tx_buffer_bit_counter))));
       radio_tx_buffer_bit_counter += 1;
       if (radio_tx_buffer_bit_counter > 7)
       {
         radio_tx_buffer_bit_counter = 0;
         radio_tx_buffer_gen_counter += 1;
       }
    }
}
//======================================== Convolutional encoding
============================================
```

```
void Convolutional_encode()
{
    header_table_counter_x= 0;
    header_table_counter_y = 0;
    descramble_poly = B00000000;
    header_table_bit_counter = 0;
    byte uhbc = 0;

    for (decoded_header_counter = 1; decoded_header_counter <= 41;
decoded_header_counter++)
    {
      for (uhbc = 0; uhbc <= 7; uhbc++)
      {
        descramble_poly = descramble_poly << 1;
        if (bitRead(decoded_header[decoded_header_counter],uhbc) == HIGH)
bitSet(descramble_poly,0); else bitClear(descramble_poly,0);
        CONVOL_ENCODE();
        if(header_table_bit_counter > 7)
        {
          header_table_bit_counter = 0;
          header_table_counter_y += 1;
          if(header_table_counter_y > 2)
          {
            header_table_counter_y = 0;
            header_table_counter_x += 1;
          }
        }
      }
    }

    // Add 2 extra 0's
    descramble_poly = descramble_poly << 1;
    bitClear(descramble_poly,0);
    CONVOL_ENCODE();

    descramble_poly = descramble_poly << 1;
    bitClear(descramble_poly,0);
    CONVOL_ENCODE();


}
//======================================== Encode routine
=====================================================
void CONVOL_ENCODE()
{

     bitWrite(header_table[header_table_counter_x][header_table_counter_y],
header_table_bit_counter,(bitRead(descramble_poly,0) ^ bitRead(descramble_poly,1) ^
bitRead(descramble_poly,2) ) );
     header_table_bit_counter += 1;
     bitWrite(header_table[header_table_counter_x][header_table_counter_y],
header_table_bit_counter,(bitRead(descramble_poly,0) ^ bitRead(descramble_poly,2) ) );
     header_table_bit_counter += 1;
}
//======================================== Convolutional decoding
=============================================
void Convolutional_decode()
{
    header_table_counter_x= 0;
    header_table_counter_y = 0;
    descramble_poly = B00000000;
```

```
    for (decoded_header_counter = 0; decoded_header_counter <= 40;
decoded_header_counter++)
    {
      decoded_header_byte = B00000000;

      if (bitRead(header_table[header_table_counter_x][header_table_counter_y],6) ==
HIGH) bitSet(descramble_poly, 7); else bitClear(descramble_poly, 7);
      CONVOL_DECODE();
      if (bitRead(header_table[header_table_counter_x][header_table_counter_y],4) ==
HIGH) bitSet(descramble_poly, 7); else bitClear(descramble_poly, 7);
      CONVOL_DECODE();
      if (bitRead(header_table[header_table_counter_x][header_table_counter_y],2) ==
HIGH) bitSet(descramble_poly, 7); else bitClear(descramble_poly, 7);
      CONVOL_DECODE();
      if (bitRead(header_table[header_table_counter_x][header_table_counter_y],0) ==
HIGH) bitSet(descramble_poly, 7); else bitClear(descramble_poly, 7);
      CONVOL_DECODE();

      header_table_counter_y += 1;
      if (header_table_counter_y > 2)
      {
        header_table_counter_y = 0;
        header_table_counter_x += 1;
      }

      if (bitRead(header_table[header_table_counter_x][header_table_counter_y],6) ==
HIGH) bitSet(descramble_poly, 7); else bitClear(descramble_poly, 7);
      CONVOL_DECODE();
      if (bitRead(header_table[header_table_counter_x][header_table_counter_y],4) ==
HIGH) bitSet(descramble_poly, 7); else bitClear(descramble_poly, 7);
      CONVOL_DECODE();
      if (bitRead(header_table[header_table_counter_x][header_table_counter_y],2) ==
HIGH) bitSet(descramble_poly, 7); else bitClear(descramble_poly, 7);
      CONVOL_DECODE();
      if (bitRead(header_table[header_table_counter_x][header_table_counter_y],0) ==
HIGH) bitSet(descramble_poly, 7); else bitClear(descramble_poly, 7);
      CONVOL_DECODE();

      header_table_counter_y += 1;
      if (header_table_counter_y > 2)
      {
        header_table_counter_y = 0;
        header_table_counter_x += 1;
      }
      decoded_header[decoded_header_counter] = decoded_header_byte;
      }
      rx_status = 4;




      // Send decoded header
      serial_out[2] = 0x2F;
      serial_out[3] = 0x00;
      serial_out[4] = 0x17;
      serial_out[5] = transmission_counter;
      serial_out[6] = packet_counter;
      serial_out[7] = 0x00;
      serial_out[8] = 0x00;
      for (decoded_header_counter = 0; decoded_header_counter <= 40;
decoded_header_counter++)
```

```
      {
        serial_out[9 + decoded_header_counter] =
decoded_header[decoded_header_counter];
      }

      serial_length = 50;
      Send_Serial();
}
//======================================== Read radio header
================================================
void Read_radio_header()
{
    if (bitRead(descramble_poly, 3) == HIGH && bitRead(descramble_poly, 6) == HIGH)
    {
      descramble_poly = descramble_poly << 1;
      bitClear(descramble_poly, 0);
    }
    else if (bitRead(descramble_poly, 3) == LOW && bitRead(descramble_poly, 6) == LOW)
    {
      descramble_poly = descramble_poly << 1;
      bitClear(descramble_poly, 0);
    }
    else
    {
      descramble_poly = descramble_poly << 1;
      bitSet(descramble_poly, 0);
    }

    if (bitRead(temp_shift_buffer[10], 0) == HIGH && bitRead(descramble_poly, 0) ==
HIGH)
    {
      bitClear(header_table[header_table_counter_x][header_table_counter_y],
header_table_bit_counter);
    }
    else if (bitRead(temp_shift_buffer[10], 0) == LOW && bitRead(descramble_poly, 0)
== LOW)
    {
      bitClear(header_table[header_table_counter_x][header_table_counter_y],
header_table_bit_counter);
    }
    else
    {
      bitSet(header_table[header_table_counter_x][header_table_counter_y],
header_table_bit_counter);
    }

    header_table_counter_x += 1;

    if (header_table_counter_y == 0 && header_table_counter_x > 27)
    {
      header_table_counter_x = 0;
      header_table_bit_counter -= 1;
      if (header_table_bit_counter <0)
      {
        header_table_bit_counter = 7;
        header_table_counter_y += 1;
      }
    }

    if (header_table_counter_y == 1 && header_table_counter_x > 27)
    {
      header_table_counter_x = 0;
```

```
      header_table_bit_counter -= 1;
    }

     if (header_table_counter_y == 1 && header_table_counter_x == 27 &&
header_table_bit_counter < 4)
    {
      header_table_counter_x = 0;
      header_table_bit_counter -= 1;
    }
     if (header_table_bit_counter <0)
      {
        header_table_bit_counter = 7;
        header_table_counter_y += 1;
      }

    if (header_table_counter_y == 2 && header_table_counter_x > 26)
    {
      header_table_counter_x = 0;
      header_table_bit_counter -= 1;
      if (header_table_bit_counter <0)
      {
        rx_status =3;
      }
    }
}
//======================================== Wait for frame sync
================================================
void Wait_for_frame_sync()
{
  if ( temp_shift_buffer[14] == B10101010
    && temp_shift_buffer[13] == B10101010
    && temp_shift_buffer[12] == B01101110
    && temp_shift_buffer[11] == B00001010)
    {
      rx_status = 2;
      header_table_counter_y = 0;
      header_table_counter_x = 0;
      header_table_bit_counter = 7;
      descramble_poly = B11111111;

      // Send syncword detected
      packet_counter =0;
      transmission_counter +=1;

      serial_out[2] = 0x03;
      serial_out[3] = 0x00;
      serial_out[4] = 0x15;
      serial_out[5] = 0x00;
      serial_out[6] = 0x00;

      serial_length = 6;
      Send_Serial();

      serial_out[2] = 0x03;
      serial_out[3] = 0x00;
      serial_out[4] = 0x16;
      serial_out[5] = transmission_counter;
      serial_out[6] = packet_counter;

      serial_length = 6;
      Send_Serial();
    }
```

```
}
//======================================== Wait for Preamble
================================================
void Wait_for_preamble()
{
   if ( temp_shift_buffer[14] == B10101010
    && temp_shift_buffer[13] == B10101010
    && temp_shift_buffer[12] == B10101010
    && temp_shift_buffer[11] == B10101010)
    {
      rx_status = 1;
      bitSet(status_flags_hi, 0);        // Set receiving status
      digitalWrite(pinPLLACQ, LOW);
    }
}
//======================================== Convolutional decoder
============================================
void CONVOL_DECODE()
{
  if (bitRead(descramble_poly, 7) == HIGH && bitRead(descramble_poly, 2) == HIGH)
  {
    bitClear(descramble_poly, 0);
    decoded_header_byte = decoded_header_byte >> 1;
    bitClear(decoded_header_byte, 7);
    descramble_poly = descramble_poly << 1;
  }

  else if (bitRead(descramble_poly, 7) == LOW && bitRead(descramble_poly, 2) == LOW)

  {
    bitClear(descramble_poly, 0);
    decoded_header_byte = decoded_header_byte >> 1;
    bitClear(decoded_header_byte, 7);
    descramble_poly = descramble_poly << 1;
  }

  else

  {
    bitSet(descramble_poly, 0);
    decoded_header_byte = decoded_header_byte >> 1;
    bitSet(decoded_header_byte, 7);
    descramble_poly = descramble_poly << 1;
  }

}
//============================ Rotate temp_shift_buffer
============================================================
void RX_buffer_shift()
{
    temp_shift_buffer_bit_counter += 1;                                        //
Increment curent possition

    if (temp_shift_buffer_bit_counter > 7)                                     //
Check if byte is full
    {
      temp_shift_buffer_bit_counter = 0;                                       // If
full clear bit counter
      temp_shift_buffer[0] = radio_tx_buffer[temp_shift_buffer_counter];   //!
// Read byte from ringbuffer
      temp_shift_buffer_counter -= 1;                                          //
Relocate ringbuffer
```

```
      if (temp_shift_buffer_counter == 0) temp_shift_buffer_counter =
radio_rx_buf_len;        // Reloop ringbuffer
    }
                                                                                //
Pleace byte in front of
                                                                                // RX
shift buffer
    for (int RX_buffer_shift_counter = 22; RX_buffer_shift_counter >= 0;
RX_buffer_shift_counter--)
    {
      temp_shift_buffer[RX_buffer_shift_counter] =
temp_shift_buffer[RX_buffer_shift_counter] >> 1;
      bitWrite(temp_shift_buffer[RX_buffer_shift_counter], 7,
bitRead(temp_shift_buffer[RX_buffer_shift_counter-1],0));
    }

}
//====================================== Interrupt routine TX clk from CMX589
================================
void TX_589()
{

  time_out_timer += 1;

    TX_delay_counter -= 1;

   // Send Preamble
   if(stream_status_modem == 4 || stream_status_modem == 5)
   {

     status_tx_bit_counter = 0;
     preamble_counter -= 1;

     if(digitalRead(pinTXD) == HIGH)
     {
       digitalWrite(pinTXD, LOW);
     }
     else
     {
       digitalWrite(pinTXD, HIGH);
     }

     // Stop preamble, stops with 1, last 01010 is added to begin of framesync to get
a full byte
    if(stream_status_modem ==5)
    {

      if(tx_inverse == false)
      {
        if(digitalRead(pinTXD) == HIGH) stream_status_modem = 6;
      }
      else
      {
        if(digitalRead(pinTXD) == LOW) stream_status_modem = 6;
      }
    }
   }



    // Send Voice / Data
```

```
   if(stream_status_modem == 7 || stream_status_modem == 8)
   {


     status_tx_bit_counter += 1;
     if (status_tx_bit_counter >= 96)
   {
#if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
 //   Serial1.print(status_unsend_frames);
 //   Serial1.write(13);
 //   Serial1.write(10);
#endif
     status_tx_bit_counter = 0;
     status_total_packets_send += 1;
     if(status_total_packets_send > 255) status_total_packets_send = 0;
   }



if(bitRead(radio_tx_buffer[radio_tx_buffer_out_counter],radio_tx_buffer_bit_counter)
== HIGH)
    {
      if(tx_inverse == true)
      {
        digitalWrite(pinTXD, LOW);
      }
      else
      {
        digitalWrite(pinTXD, HIGH);
      }
    }
    else
    {
      if(tx_inverse == true)
      {
        digitalWrite(pinTXD, HIGH);
      }
      else
      {
        digitalWrite(pinTXD, LOW);
      }
    }

    radio_tx_buffer_bit_counter += 1;

    if (radio_tx_buffer_bit_counter == 8)
    {

      radio_tx_buffer_bit_counter = 0;
      radio_tx_buffer_out_counter += 1;
      if (radio_tx_buffer_out_counter > radio_tx_buf_len)
      {
        radio_tx_buffer_out_counter = 1;
      }
    }


    if(stream_status_modem == 8)
    {
      if(radio_tx_buffer_in_counter == radio_tx_buffer_out_counter)
stream_status_modem = 9;
```

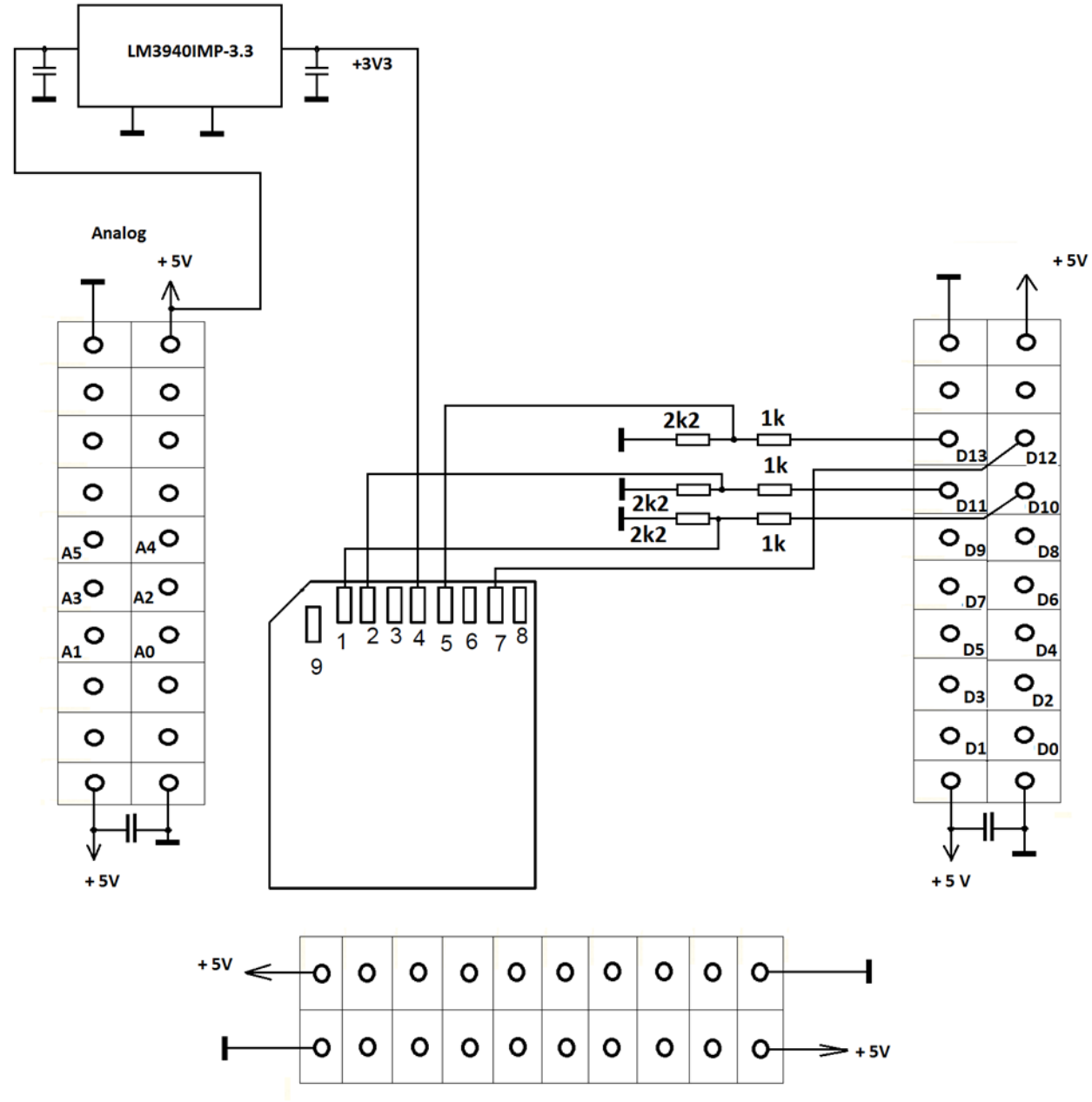
```
    }
  }

  tx_int_flag = true;
}
//======================================== Interrupt routine RX clk from CMX589
================================
void RX_589()
{

  time_out_timer += 1;

  if (rx_inverse == false)                                          // Check data
polarity
  {
  bitWrite(input_rx_buffer, 7, digitalRead(pinRXD));                // Read RX
data pin normal
  }
  else
  {
  bitWrite(input_rx_buffer, 7, !digitalRead(pinRXD));               // Read RX
data pin inverse
  }
  input_rx_buffer_bit_counter += 1;                                 // Increment
bit counter

  if (input_rx_buffer_bit_counter == 8 )                            // If byte is
full copy byte to ringbuffer
  {
    radio_tx_buffer[radio_rx_buffer_counter] = input_rx_buffer;    //!     // Copy
byte to ringbuffer
    radio_rx_buffer_counter -= 1;                                   // Rotate
ringbuffer
    if (radio_rx_buffer_counter == 0) radio_rx_buffer_counter = radio_rx_buf_len;
// Loop buffer around if needed
    input_rx_buffer_bit_counter = 0;                                //
    }
  else
    {                                                               // If byte is
not full
    input_rx_buffer = input_rx_buffer >> 1;                         // Rotate
input buffer
  }

  rx_int_flag = true;                                               // Set
rx_int_flag for general purposes.

}
//=============================================================================
==============================
void ADF7021_ini()
{

       AD7021_control_byte = 0x00575011; // 1
       Send_AD7021_control();

       AD7021_control_byte = 0x281500A3; // 3 RX rx werkend test commen
       Send_AD7021_control();

       AD7021_control_byte = 0x19614080; // 0
       Send_AD7021_control();
```

```
       AD7021_control_byte = 0x00003155; // 5
       Send_AD7021_control();

       AD7021_control_byte = 0x0037B014; // 4  test common
       Send_AD7021_control();

       AD7021_control_byte = 0x0115D092; // 2 test common
       Send_AD7021_control();

       AD7021_control_byte = 0x000E000F; // 15
       Send_AD7021_control();

       AD7021_control_byte = 0x050972C6; // 6
       Send_AD7021_control();

       AD7021_control_byte = 0x000231E9; // 9
       Send_AD7021_control();

       AD7021_control_byte = 0x3296355A; // 10
       Send_AD7021_control();

       AD7021_control_byte = 0x0000003B; // 11
       Send_AD7021_control();

       AD7021_control_byte = 0x0000010C; // 12
       Send_AD7021_control();

       AD7021_control_byte = 0x0000000D; // 13
       Send_AD7021_control();
}
//=============================================================================================
==================================
void ADF7021_tx()
{
       AD7021_control_byte = 0x0075D092; // 2 tx
       Send_AD7021_control();

      AD7021_control_byte = 0x2B148123; // 3 tx
       Send_AD7021_control();

       AD7021_control_byte = 0x00593814; // 4 tx
       Send_AD7021_control();

       AD7021_control_byte = 0x11615550; // Set tx
       Send_AD7021_control();
}

//=============================================================================================
==================================
void ADF7021_rx()
{
       AD7021_control_byte = 0x00003155; // 5
       Send_AD7021_control();

      AD7021_control_byte = 0x0115D092; // 2 RX
       Send_AD7021_control();

      AD7021_control_byte = 0x2B1500A3; // 3 RX
       Send_AD7021_control();

        AD7021_control_byte = 0x0037B014; // 4 RX
       Send_AD7021_control();
```

```
        AD7021_control_byte = 0x19614080; // set rx
        Send_AD7021_control();
}
```

5. SD karta

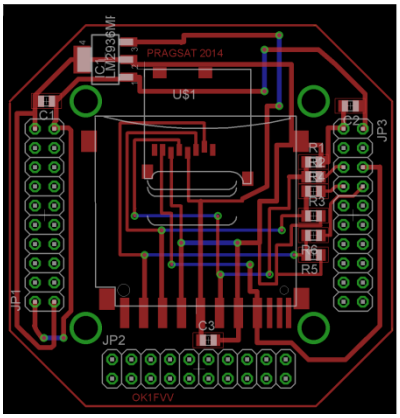
SD karta komunikuje pomocí protokolu SPI. Zapojení desky s SD kartou je:

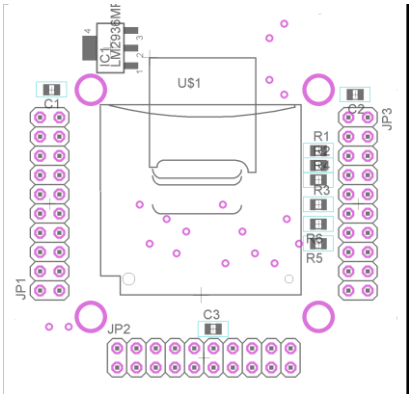


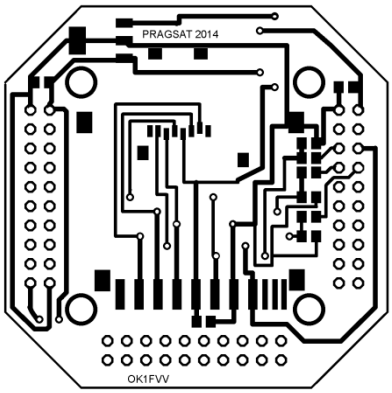
OK1FVV

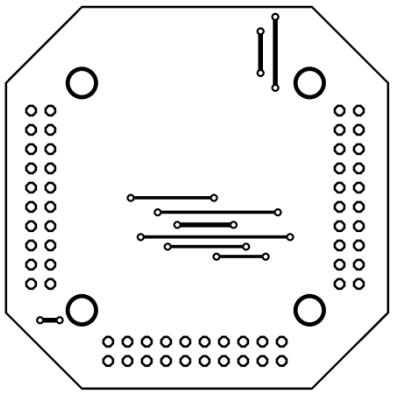
Destička umožňuje osazení buď slotem pro SD kartu nebo slotem pro SD mikro kartu. Osadil jsem slotem SD. Zapojení je jednoduché, jde o SPI komunikaci s Arduinem. Protože SD karta je napájena 3V3, je na desctičce i zdroj tohoto napětí. Dále jsou zde tři odporové děliče snižující úroveň SPI

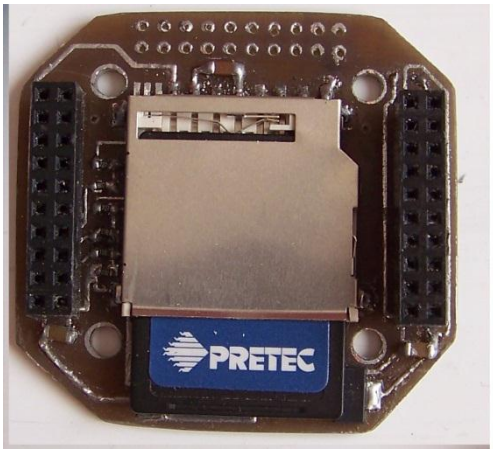
signálů z Arduina z V na 3V3 pro SD kartu. Výběr SD karty CS je připojen k A10, což musí být respektováno v obslužném sw.












```
/*
  SD card read/write

 This example shows how to read and write data to and from an SD card file
 The circuit:
 * SD card attached to SPI bus as follows:
 ** MOSI - pin 11
 ** MISO - pin 12
 ** CLK - pin 13
```

```
 ** CS - pin 4

 */

#include <SD.h>
const int chipSelect = 10;
File myFile;

void setup()
{
 // Open serial communications and wait for port to open:
  Serial.begin(9600);
   while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }


  Serial.print("Initializing SD card...");
  // On the Ethernet Shield, CS is pin 4. It's set as an output by default.
  // Note that even if it's not used as the CS pin, the hardware SS pin
  // (10 on most Arduino boards, 53 on the Mega) must be left as an output
  // or the SD library functions will not work.
   pinMode(10, OUTPUT);

  //if (!SD.begin(4)) {
    if (!SD.begin(10)) {
    Serial.println("initialization failed!");
    return;
  }
  Serial.println("initialization done.");

  // open the file. note that only one file can be open at a time,
  // so you have to close this one before opening another.
  myFile = SD.open("test.txt", FILE_WRITE);

  // if the file opened okay, write to it:
  if (myFile) {
    Serial.print("Writing to test.txt...");
    myFile.println("testing 1, 2, 3.,4,5 ahoj");
       // close the file:
    myFile.close();
    Serial.println("done.");
  } else {
    // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }

  // re-open the file for reading:
  myFile = SD.open("test.txt");
  if (myFile) {
    Serial.println("test.txt:");

    // read from the file until there's nothing else in it:
    while (myFile.available()) {
       Serial.write(myFile.read());
    }
    // close the file:
    myFile.close();
  } else {
       // if the file didn't open, print an error:
    Serial.println("error opening test.txt");
  }
```

```
}

void loop()
{
       // nothing happens after setup
}
```

Soubor **SD.h**

```
/*

 SD - a slightly more friendly wrapper for sdfatlib

 This library aims to expose a subset of SD card functionality

 */

#ifndef __SD_H__
#define __SD_H__

#include <Arduino.h>

#include <utility/SdFat.h>
#include <utility/SdFatUtil.h>

#define FILE_READ O_READ
#define FILE_WRITE (O_READ | O_WRITE | O_CREAT)

class File : public Stream {
 private:
  char _name[13]; // our name
  SdFile *_file;  // underlying file pointer

public:
  File(SdFile f, const char *name);     // wraps an underlying SdFile
  File(void);      // 'empty' constructor
  ~File(void);     // destructor
  virtual size_t write(uint8_t);
  virtual size_t write(const uint8_t *buf, size_t size);
  virtual int read();
  virtual int peek();
  virtual int available();
  virtual void flush();
  int read(void *buf, uint16_t nbyte);
  boolean seek(uint32_t pos);
  uint32_t position();
  uint32_t size();
  void close();
  operator bool();
  char * name();

  boolean isDirectory(void);
  File openNextFile(uint8_t mode = O_RDONLY);
  void rewindDirectory(void);

  using Print::write;
};
```

```
class SDClass {

private:
  // These are required for initialisation and use of sdfatlib
  Sd2Card card;
  SdVolume volume;
  SdFile root;

  // my quick&dirty iterator, should be replaced
  SdFile getParentDir(const char *filepath, int *indx);
public:
  // This needs to be called to set up the connection to the SD card
  // before other methods are used.
  boolean begin(uint8_t csPin = SD_CHIP_SELECT_PIN);

  // Open the specified file/directory with the supplied mode (e.g. read or
  // write, etc). Returns a File object for interacting with the file.
  // Note that currently only one file can be open at a time.
  File open(const char *filename, uint8_t mode = FILE_READ);

  // Methods to determine if the requested file path exists.
  boolean exists(char *filepath);

  // Create the requested directory heirarchy--if intermediate directories
  // do not exist they will be created.
  boolean mkdir(char *filepath);

  // Delete the file.
  boolean remove(char *filepath);

  boolean rmdir(char *filepath);

private:

  // This is used to determine the mode used to open a file
  // it's here because it's the easiest place to pass the
  // information through the directory walking function. But
  // it's probably not the best place for it.
  // It shouldn't be set directly--it is set via the parameters to `open`.
  int fileOpenMode;

  friend class File;
  friend boolean callback_openPath(SdFile&, char *, boolean, void *);
};

extern SDClass SD;

#endif
```

Soubor **SD.cpp**

```
/*

 SD - a slightly more friendly wrapper for sdfatlib
```

```
 */


#include "SD.h"

// Used by `getNextPathComponent`
#define MAX_COMPONENT_LEN 12 // What is max length?
#define PATH_COMPONENT_BUFFER_LEN MAX_COMPONENT_LEN+1

bool getNextPathComponent(char *path, unsigned int *p_offset,
                          char *buffer) {
  /*

    Parse individual path components from a path.

      e.g. after repeated calls '/foo/bar/baz' will be split
           into 'foo', 'bar', 'baz'.

    This is similar to `strtok()` but copies the component into the
    supplied buffer rather than modifying the original string.


    `buffer` needs to be PATH_COMPONENT_BUFFER_LEN in size.

    `p_offset` needs to point to an integer of the offset at
    which the previous path component finished.

    Returns `true` if more components remain.

    Returns `false` if this is the last component.
      (This means path ended with 'foo' or 'foo/'.)

   */

  // TODO: Have buffer local to this function, so we know it's the
  //       correct length?

  int bufferOffset = 0;

  int offset = *p_offset;

  // Skip root or other separator
  if (path[offset] == '/') {
    offset++;
  }

  // Copy the next next path segment
  while (bufferOffset < MAX_COMPONENT_LEN
         && (path[offset] != '/')
         && (path[offset] != '\0')) {
    buffer[bufferOffset++] = path[offset++];
  }

  buffer[bufferOffset] = '\0';

  // Skip trailing separator so we can determine if this
  // is the last component in the path or not.
  if (path[offset] == '/') {
    offset++;
  }

  *p_offset = offset;
```

```
  return (path[offset] != '\0');
}



boolean walkPath(char *filepath, SdFile& parentDir,
               boolean (*callback)(SdFile& parentDir,
                                   char *filePathComponent,
                                   boolean isLastComponent,
                                   void *object),
               void *object = NULL) {
  /*

     When given a file path (and parent directory--normally root),
     this function traverses the directories in the path and at each
     level calls the supplied callback function while also providing
     the supplied object for context if required.

       e.g. given the path '/foo/bar/baz'
            the callback would be called at the equivalent of
           '/foo', '/foo/bar' and '/foo/bar/baz'.

     The implementation swaps between two different directory/file
     handles as it traverses the directories and does not use recursion
     in an attempt to use memory efficiently.

     If a callback wishes to stop the directory traversal it should
     return false--in this case the function will stop the traversal,
     tidy up and return false.

     If a directory path doesn't exist at some point this function will
     also return false and not subsequently call the callback.

     If a directory path specified is complete, valid and the callback
     did not indicate the traversal should be interrupted then this
     function will return true.

   */


  SdFile subfile1;
  SdFile subfile2;

  char buffer[PATH_COMPONENT_BUFFER_LEN];

  unsigned int offset = 0;

  SdFile *p_parent;
  SdFile *p_child;

  SdFile *p_tmp_sdfile;

  p_child = &subfile1;

  p_parent = &parentDir;

  while (true) {

    boolean moreComponents = getNextPathComponent(filepath, &offset, buffer);

    boolean shouldContinue = callback((*p_parent), buffer, !moreComponents, object);
```

```
    if (!shouldContinue) {
      // TODO: Don't repeat this code?
      // If it's one we've created then we
      // don't need the parent handle anymore.
      if (p_parent != &parentDir) {
        (*p_parent).close();
      }
      return false;
    }

    if (!moreComponents) {
      break;
    }

    boolean exists = (*p_child).open(*p_parent, buffer, O_RDONLY);

    // If it's one we've created then we
    // don't need the parent handle anymore.
    if (p_parent != &parentDir) {
      (*p_parent).close();
    }

    // Handle case when it doesn't exist and we can't continue...
    if (exists) {
      // We alternate between two file handles as we go down
      // the path.
      if (p_parent == &parentDir) {
        p_parent = &subfile2;
      }

      p_tmp_sdfile = p_parent;
      p_parent = p_child;
      p_child = p_tmp_sdfile;
    } else {
      return false;
    }
  }

  if (p_parent != &parentDir) {
    (*p_parent).close(); // TODO: Return/ handle different?
  }

  return true;
}



/*

   The callbacks used to implement various functionality follow.

   Each callback is supplied with a parent directory handle,
   character string with the name of the current file path component,
   a flag indicating if this component is the last in the path and
   a pointer to an arbitrary object used for context.

 */

boolean callback_pathExists(SdFile& parentDir, char *filePathComponent,
                            boolean isLastComponent, void *object) {
  /*
```

```
    Callback used to determine if a file/directory exists in parent
    directory.

    Returns true if file path exists.

  */
  SdFile child;

  boolean exists = child.open(parentDir, filePathComponent, O_RDONLY);

  if (exists) {
     child.close();
  }

  return exists;
}



boolean callback_makeDirPath(SdFile& parentDir, char *filePathComponent,
                             boolean isLastComponent, void *object) {
  /*

    Callback used to create a directory in the parent directory if
    it does not already exist.

    Returns true if a directory was created or it already existed.

  */
  boolean result = false;
  SdFile child;

  result = callback_pathExists(parentDir, filePathComponent, isLastComponent, object);
  if (!result) {
    result = child.makeDir(parentDir, filePathComponent);
  }

  return result;
}


  /*

boolean callback_openPath(SdFile& parentDir, char *filePathComponent,
                          boolean isLastComponent, void *object) {

    Callback used to open a file specified by a filepath that may
    specify one or more directories above it.

    Expects the context object to be an instance of `SDClass` and
    will use the `file` property of the instance to open the requested
    file/directory with the associated file open mode property.

    Always returns true if the directory traversal hasn't reached the
    bottom of the directory heirarchy.

    Returns false once the file has been opened--to prevent the traversal
    from descending further. (This may be unnecessary.)

  if (isLastComponent) {
    SDClass *p_SD = static_cast<SDClass*>(object);
```

```
    p_SD->file.open(parentDir, filePathComponent, p_SD->fileOpenMode);
    if (p_SD->fileOpenMode == FILE_WRITE) {
      p_SD->file.seekSet(p_SD->file.fileSize());
    }
    // TODO: Return file open result?
    return false;
  }
  return true;
}
  */



boolean callback_remove(SdFile& parentDir, char *filePathComponent,
                        boolean isLastComponent, void *object) {
  if (isLastComponent) {
    return SdFile::remove(parentDir, filePathComponent);
  }
  return true;
}

boolean callback_rmdir(SdFile& parentDir, char *filePathComponent,
                       boolean isLastComponent, void *object) {
  if (isLastComponent) {
    SdFile f;
    if (!f.open(parentDir, filePathComponent, O_READ)) return false;
    return f.rmDir();
  }
  return true;
}



/* Implementation of class used to create `SDCard` object. */



boolean SDClass::begin(uint8_t csPin) {
  /*

    Performs the initialisation required by the sdfatlib library.

    Return true if initialization succeeds, false otherwise.

   */
  return card.init(SPI_HALF_SPEED, csPin) &&
         volume.init(card) &&
         root.openRoot(volume);
}



// this little helper is used to traverse paths
SdFile SDClass::getParentDir(const char *filepath, int *index) {
  // get parent directory
  SdFile d1 = root; // start with the mostparent, root!
  SdFile d2;

  // we'll use the pointers to swap between the two objects
  SdFile *parent = &d1;
  SdFile *subdir = &d2;
```

```
  const char *origpath = filepath;

  while (strchr(filepath, '/')) {

    // get rid of leading /'s
    if (filepath[0] == '/') {
      filepath++;
      continue;
    }

    if (! strchr(filepath, '/')) {
      // it was in the root directory, so leave now
      break;
    }

    // extract just the name of the next subdirectory
    uint8_t idx = strchr(filepath, '/') - filepath;
    if (idx > 12)
      idx = 12;    // dont let them specify long names
    char subdirname[13];
    strncpy(subdirname, filepath, idx);
    subdirname[idx] = 0;

    // close the subdir (we reuse them) if open
    subdir->close();
    if (! subdir->open(parent, subdirname, O_READ)) {
      // failed to open one of the subdirectories
      return SdFile();
    }
    // move forward to the next subdirectory
    filepath += idx;

    // we reuse the objects, close it.
    parent->close();

    // swap the pointers
    SdFile *t = parent;
    parent = subdir;
    subdir = t;
  }

  *index = (int)(filepath - origpath);
  // parent is now the parent diretory of the file!
  return *parent;
}


File SDClass::open(const char *filepath, uint8_t mode) {
  /*

     Open the supplied file path for reading or writing.

     The file content can be accessed via the `file` property of
     the `SDClass` object--this property is currently
     a standard `SdFile` object from `sdfatlib`.

     Defaults to read only.

     If `write` is true, default action (when `append` is true) is to
     append data to the end of the file.

     If `append` is false then the file will be truncated first.
```

```
    If the file does not exist and it is opened for writing the file
    will be created.

    An attempt to open a file for reading that does not exist is an
    error.

   */

  int pathidx;

  // do the interative search
  SdFile parentdir = getParentDir(filepath, &pathidx);
  // no more subdirs!

  filepath += pathidx;

  if (! filepath[0]) {
    // it was the directory itself!
    return File(parentdir, "/");
  }

  // Open the file itself
  SdFile file;

  // failed to open a subdir!
  if (!parentdir.isOpen())
    return File();

  // there is a special case for the Root directory since its a static dir
  if (parentdir.isRoot()) {
    if ( ! file.open(SD.root, filepath, mode)) {
      // failed to open the file :(
      return File();
    }
    // dont close the root!
  } else {
    if ( ! file.open(parentdir, filepath, mode)) {
      return File();
    }
    // close the parent
    parentdir.close();
  }

  if (mode & (O_APPEND | O_WRITE))
    file.seekSet(file.fileSize());
  return File(file, filepath);
}


/*
File SDClass::open(char *filepath, uint8_t mode) {
  //

    Open the supplied file path for reading or writing.

    The file content can be accessed via the `file` property of
    the `SDClass` object--this property is currently
    a standard `SdFile` object from `sdfatlib`.

    Defaults to read only.
```

```
     If `write` is true, default action (when `append` is true) is to
     append data to the end of the file.

     If `append` is false then the file will be truncated first.

     If the file does not exist and it is opened for writing the file
     will be created.

     An attempt to open a file for reading that does not exist is an
     error.

   //

  // TODO: Allow for read&write? (Possibly not, as it requires seek.)

  fileOpenMode = mode;
  walkPath(filepath, root, callback_openPath, this);

  return File();

}
*/


//boolean SDClass::close() {
//  /*
//
//    Closes the file opened by the `open` method.
//
//   */
//  file.close();
//}


boolean SDClass::exists(char *filepath) {
  /*

     Returns true if the supplied file path exists.

   */
  return walkPath(filepath, root, callback_pathExists);
}


//boolean SDClass::exists(char *filepath, SdFile& parentDir) {
//  /*
//
//     Returns true if the supplied file path rooted at `parentDir`
//     exists.
//
//   */
//  return walkPath(filepath, parentDir, callback_pathExists);
//}


boolean SDClass::mkdir(char *filepath) {
  /*

    Makes a single directory or a heirarchy of directories.

    A rough equivalent to `mkdir -p`.
```

```
   */
  return walkPath(filepath, root, callback_makeDirPath);
}

boolean SDClass::rmdir(char *filepath) {
  /*

    Makes a single directory or a heirarchy of directories.

    A rough equivalent to `mkdir -p`.

   */
  return walkPath(filepath, root, callback_rmdir);
}

boolean SDClass::remove(char *filepath) {
  return walkPath(filepath, root, callback_remove);
}


// allows you to recurse into a directory
File File::openNextFile(uint8_t mode) {
  dir_t p;

  //Serial.print("\t\treading dir...");
  while (_file->readDir(&p) > 0) {

    // done if past last used entry
    if (p.name[0] == DIR_NAME_FREE) {
      //Serial.println("end");
      return File();
    }

    // skip deleted entry and entries for . and  ..
    if (p.name[0] == DIR_NAME_DELETED || p.name[0] == '.') {
      //Serial.println("dots");
      continue;
    }

    // only list subdirectories and files
    if (!DIR_IS_FILE_OR_SUBDIR(&p)) {
      //Serial.println("notafile");
      continue;
    }

    // print file name with possible blank fill
    SdFile f;
    char name[13];
    _file->dirName(p, name);
    //Serial.print("try to open file ");
    //Serial.println(name);

    if (f.open(_file, name, mode)) {
      //Serial.println("OK!");
      return File(f, name);
    } else {
      //Serial.println("ugh");
      return File();
    }
  }

  //Serial.println("nothing");
```

```
  return File();
}

void File::rewindDirectory(void) {
  if (isDirectory())
    _file->rewind();
}

SDClass SD;
```

6. Softwarové vybavení

6.1 firmware Cansatu

Pokud bychom z CanSATu přijímali naměřená data jen jako shluk čísel, byl by význam jednotlivých čísel dán jen tím, v jakém pořadí jsme příslušné číslo přijali. Musili bychom ovšem přijmout čísla všechna a současně mít jistotu že např. MCU vlivem rušení apod. začal vysílat opravdu na začátku předpokládané řady čísel apod. Proto je výhodné spolu s daty vysílat u údaje o tom co znamenají. Velice rozšířené pro tento účel je použití jazyka **XML**. Jeho nevýhodou pro naše účely je, že se kromě vlastních dat a informace o tom, co znamenají, přenáší velké množství dalších znaků.
Z těch důvodů se někdy místo XML používá formát JSON a použili jsme ho i my. JSON (JavaScript Object Notation) je univerzální textový formát pro výměnu dat. Je založený na syntaxi jazyka Javascript. Jazyk JSON je velmi jednoduchý a snadno se čte. Datové struktury zapsané v jazyce JSON je možné převést téměř do všech programovacích jazyků. Jeho syntaxe je postavena na dvou základních strukturách:

•množina párů (klíč - hodnota)
•seznam hodnot
Příklad:

```
{
  "jméno" : "Karel",
  "příjmení" : "Novák",
  "věk" : 30,
  "adresa" :
  {
    "ulice" : "Blanická 42",
    "město" : "Liptákov",
    "psč" : 12331,
    "země" : "Česká republika"
  },
  "zájmy" :
  [
    "zahrádka",
    "turistika",
    "fotografie",
    "chovatelství"
  ]
}
```

Náše využití (ukázka):

```
{
"time" : "2014/06/03 10:11:59",
"pressure" : "1016.63",
"chip_temperature" : "28.98",
"temperature" : "25.44",
"Ax" : "-0.16",
"Ay" : "2.67",
"Az" : "10.59",
"Mx" : "-19.18",
"My" : "-41.73",
"Mz" : "-83.98",
}
```

Pro zjednodušení jsem ovšem nepoužil CR LF, takže se vysílají obdobné řetězce znaků.

```
{"time" : "2014/06/03 10:11:59","pressure" : "1016.63","chip_temperature" : "28.98","temperature"
: "25.44",
"Ax" : "-0.16","Ay" : "2.67","Az" : "10.59","Mx" : "-19.18","My" : "-41.73","Mz" : "-83.98",}
```

Výsledný firmware pro ATMega328 našeho CanSATu je:

```
#include <Wire.h> // necessary, or the application won't build properly
#include <stdio.h>
#include <PCF8583.h>
#include <SD.h>
#include <LPS331.h>
#include <Serial.h>
#include <Sensor.h>
#include <LSM303_U.h>

/* Assign a unique ID to this sensor at the same time */
LSM303_Accel_Unified accel = LSM303_Accel_Unified(54321);
LSM303_Mag_Unified mag = LSM303_Mag_Unified(12345);

LPS331 ps;
const int chipSelect = 10;
File myFile;


/*************************************************************************
 *  read/write serial interface to PCF8583 RTC via I2C interface
 *
 *  Arduino analog input 5 - I2C SCL (PCF8583 pin 6)
 *  Arduino analog input 4 - I2C SDA (PCF8583 pin 5)
 *
 *  You can set the type by sending it YYMMddhhmmss;
 *  the semicolon on the end tells it you're done...
 *
 *************************************************************************/

int correct_address = 0;
```

```
PCF8583 p (0xA0);

char lm92address = 0x48; //A1 A0 are grounded
char buffer[2];
int tempcode;
float temp;

void setup(void){
  Serial.begin(9600);
  //Serial.print("booting...");
  //Serial.println(" done");
  Wire.begin();

  if (!ps.init())
  {
    //Serial.println("Failed to autodetect pressure sensor!");
    while (1);
  }

  ps.enableDefault();
//SD
  //Serial.print("Initializing SD card...");
  pinMode(10, OUTPUT);

    if (!SD.begin(10)) {
    //Serial.println("initialization failed!");
    return;
  }
  //Serial.println("initialization done.");
//accelerometer
 // Serial.println("Accelerometer Test"); Serial.println("");

  /* Initialise the sensor */
  if(!accel.begin())
  {
    /* There was a problem detecting the ADXL345 ... check your connections */
   // Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
    while(1);
  }
//magnetometr
 //Serial.println("Magnetometer Test"); Serial.println("");

  /* Initialise the sensor */
  if(!mag.begin())
  {
    /* There was a problem detecting the LSM303 ... check your connections */
    //Serial.println("Ooops, no LSM303 detected ... Check your wiring!");
    while(1);
  }
//hlavicka csv
    File dataFile = SD.open("data.csv", FILE_WRITE);
   if (dataFile) {  //
    dataFile.print("date");
    dataFile.print(";");
    dataFile.print("time");
    dataFile.print(";");
    dataFile.print("pressure");
    //dataFile.print(" mbar\ta: ");
    dataFile.print(";");
    //dataFile.print(altitude);
    dataFile.print("altitude");
    //dataFile.print(" m\tt: ");
```

```
    dataFile.print(";");
    //dataFile.print(temperature);
    dataFile.print("chip_temperature");
    //dataFile.print(" deg C");
    //dataFile.print("  ");
    dataFile.print(";");
 //teplota
    //dataFile.print(" t = ");
    //dataFile.print(temp);
    dataFile.print("temperature");
    //dataFile.print(" C");
    //dataFile.print("  ");
    dataFile.print(";");
    /* Display the results (acceleration is measured in m/s^2) */
    dataFile.print("Ax");
    //dataFile.print(event_a.acceleration.x);
    dataFile.print(";");// dataFile.print("  ");
    dataFile.print("Ay");
    //dataFile.print(event_a.acceleration.y);
    dataFile.print(";");// dataFile.print("  ");
    dataFile.print("Az");
    //dataFile.print(event_a.acceleration.z);
    dataFile.print(";");//dataFile.print("  ");
   // dataFile.print("m/s^2 ");
   // dataFile.print(" ");
   /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
    dataFile.print("Mx");
    //dataFile.print(event_m.magnetic.x);
    dataFile.print(";");//dataFile.print("  ");
    dataFile.print("My");
    //dataFile.print(event_m.magnetic.y);
    dataFile.print(";");//dataFile.print("  ");
    dataFile.print("Mz");
    //dataFile.print(event_m.magnetic.z);
    dataFile.print(";");// dataFile.print("  ");
   // dataFile.print("uT");
    dataFile.println();

    dataFile.close();              //soubor uzavre
    }

}



void loop(void){
 if(Serial.available() > 0){
       p.year= (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48)) + 2000;
       p.month = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
       p.day = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
       p.hour  = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
       p.minute = (byte) ((Serial.read() - 48) *10 +  (Serial.read() - 48));
       p.second = (byte) ((Serial.read() - 48) * 10 + (Serial.read() - 48)); // Use of
(byte) type casting and ascii math to achieve result.

       if(Serial.read() == ';'){
        // Serial.println("setting date");
        p.set_time();
       }
```

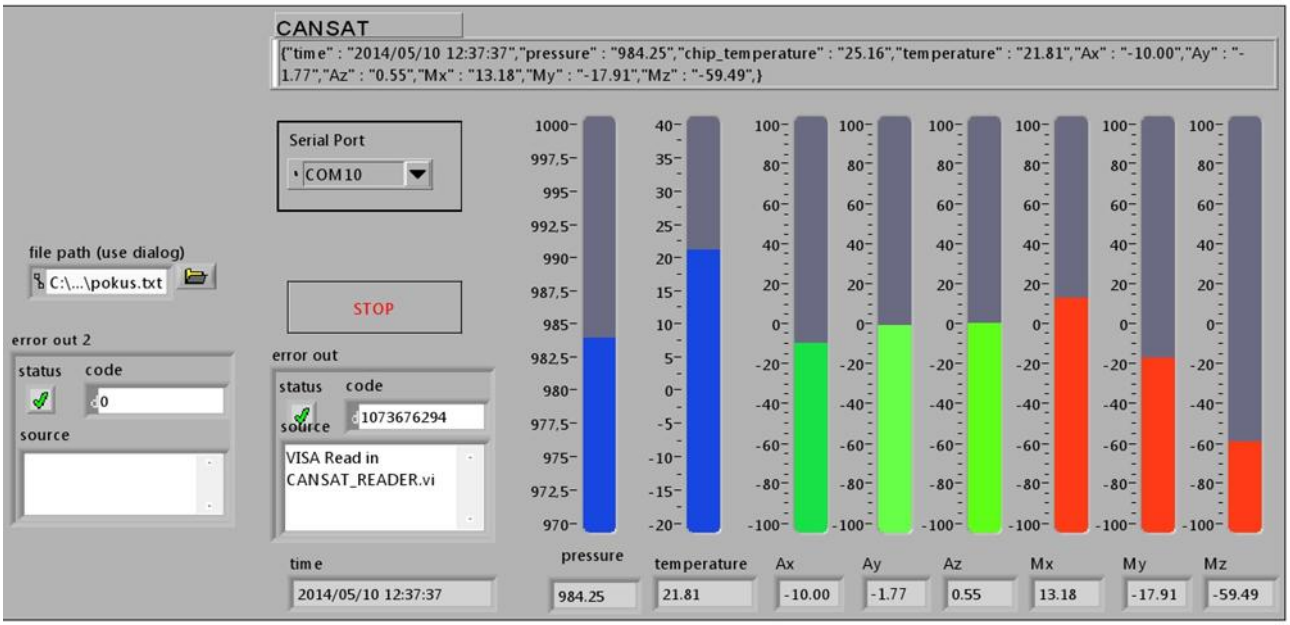
```
  }


  p.get_time();
  char time[50];
  char date[25];
  char time1[25];
  Serial.print("{\"time\" : \"");
  sprintf(time, "%02d/%02d/%02d %02d:%02d:%02d",
         p.year, p.month, p.day, p.hour, p.minute, p.second);

  sprintf(date, "%02d/%02d/%02d ",
         p.year, p.month, p.day);

  sprintf(time1, "%02d:%02d:%02d",
          p.hour, p.minute, p.second);
  //Serial.print("\"");
  Serial.print(time);
  Serial.print("\",");
   //atmosfericky tlak
  float pressure = ps.readPressureMillibars();
  float altitude = ps.pressureToAltitudeMeters(pressure);
  float temperature = ps.readTemperatureC();

  Serial.print("\"pressure\" : \"");
  Serial.print(pressure);
  //Serial.print(" mbar\ta: ");
  Serial.print("\",");
  //Serial.print(altitude);
  //Serial.print(" m\tt: ");
  Serial.print("\"chip_temperature\" : \"");
  Serial.print(temperature);
  //Serial.print(" deg C");
   Serial.print("\",");
 //teplota
 Wire.requestFrom(lm92address,2);

  if (Wire.available())

  {
    for (int i=0; i<2; i++) {

      buffer[i]= Wire.read();
    }

tempcode= ((buffer[0] << 8) + buffer[1]) >>3;
temp = tempcode * 0.0625;
 Serial.print("\"temperature\" : \"");
 //Serial.print(" t = ");
 Serial.print(temp);
 //Serial.print(" C");
 //Serial.print("  ");
 Serial.print("\",");
   /* Get a new sensor event */
  sensors_event_t event_a;
  accel.getEvent(&event_a);

  /* Display the results (acceleration is measured in m/s^2) */
  //Serial.print("Ax: ");
  Serial.print("\"Ax\" : \"");Serial.print(event_a.acceleration.x);
Serial.print("\",");//Serial.print("  ");
  //Serial.print("Ay: ");
```

```
  Serial.print("\"Ay\" : \"");Serial.print(event_a.acceleration.y);
Serial.print("\",");//Serial.print("  ");
  //Serial.print("Az: ");
  Serial.print("\"Az\" : \"");Serial.print(event_a.acceleration.z);
Serial.print("\",");//Serial.print("  ");
  //Serial.print("m/s^2 ");

  sensors_event_t event_m;
  mag.getEvent(&event_m);

  /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
  //Serial.print("Mx: ");
  Serial.print("\"Mx\" : \"");Serial.print(event_m.magnetic.x);Serial.print("\",");
//Serial.print("  ");
  //Serial.print("My: ");
  Serial.print("\"My\" : \"");Serial.print(event_m.magnetic.y);Serial.print("\",");
//Serial.print("  ");
  //Serial.print("Mz: ");
  Serial.print("\"Mz\" : \"");Serial.print(event_m.magnetic.z);Serial.print("\",");
//Serial.print("  ");
  //Serial.println("uT");
  Serial.println("}");

   delay(800);
    File dataFile = SD.open("data.csv", FILE_WRITE);
   if (dataFile) {  //
    dataFile.print(date);
    dataFile.print(";");
    dataFile.print(time1);
              //
 //   dataFile.print("  ");
    dataFile.print(";");
    //dataFile.print(" p: ");
    dataFile.print(pressure);
    //dataFile.print(" mbar\ta: ");
    dataFile.print(";");
    dataFile.print(altitude);
    //dataFile.print(" m\tt: ");
    dataFile.print(";");
    dataFile.print(temperature);
    //dataFile.print(" deg C");
    //dataFile.print("  ");
    dataFile.print(";");
 //teplota
    //dataFile.print(" t = ");
    dataFile.print(temp);
    //dataFile.print(" C");
    //dataFile.print("  ");
    dataFile.print(";");
    /* Display the results (acceleration is measured in m/s^2) */
    //dataFile.print("Ax: ");
    dataFile.print(event_a.acceleration.x);dataFile.print(";");// dataFile.print("
");
    //dataFile.print("Ay: ");
    dataFile.print(event_a.acceleration.y);dataFile.print(";");// dataFile.print("
");
    //dataFile.print("Az: ");
    dataFile.print(event_a.acceleration.z); dataFile.print(";");//dataFile.print("
");
   // dataFile.print("m/s^2 ");
   // dataFile.print(" ");
   /* Display the results (magnetic vector values are in micro-Tesla (uT)) */
```

```
   // dataFile.print("Mx: ");
    dataFile.print(event_m.magnetic.x); dataFile.print(";");//dataFile.print("  ");
    //dataFile.print("My: ");
    dataFile.print(event_m.magnetic.y);dataFile.print(";");//dataFile.print("  ");
    //dataFile.print("Mz: ");
    dataFile.print(event_m.magnetic.z);dataFile.print(";");// dataFile.print("  ");
   // dataFile.print("uT");
    dataFile.println();

    dataFile.close();              //soubor uzavre
    }
  else {
    Serial.println("error opening data.txt");
  }

}


}
```

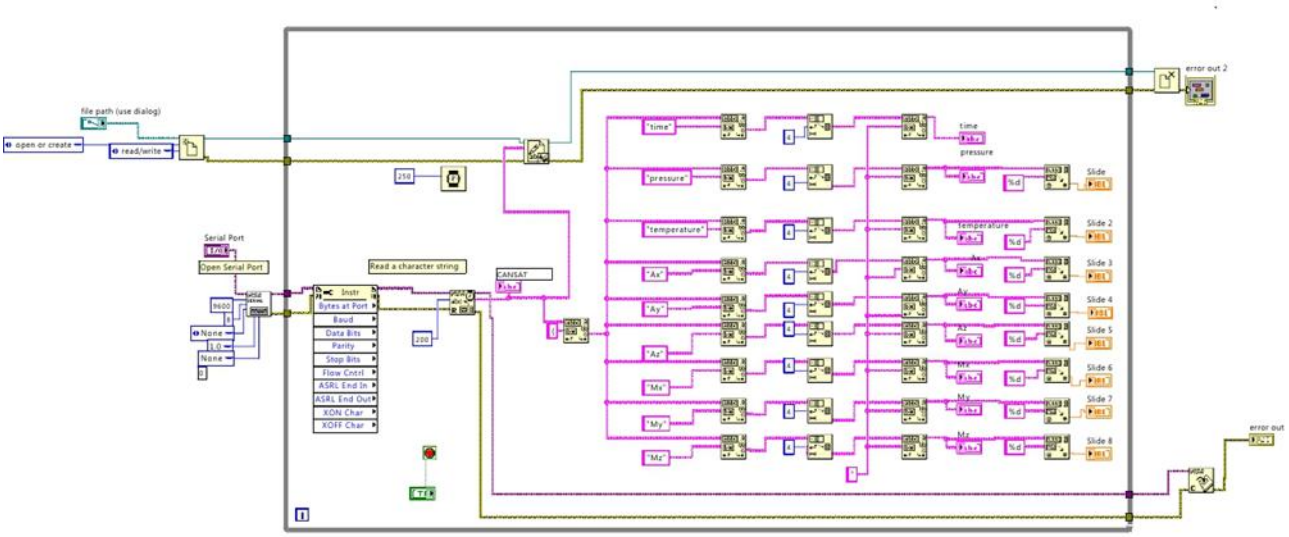
6.2 software na pozemní stanici

Použil jsem LabView Student edition. Výsledkem je :



Pod nápisem CANSAT vidíme přijatý řetězec ve formátu JSON a dále pak texboxy s hodnotami teploty, tlaku, 3D akcelerometru a 3D magnetometru a nad nimi sloupcové grafy s těmito hodnotami.

Block Diagram



Kromě toho program přijaté řetězce znaků zapisuje do souboru na PC.

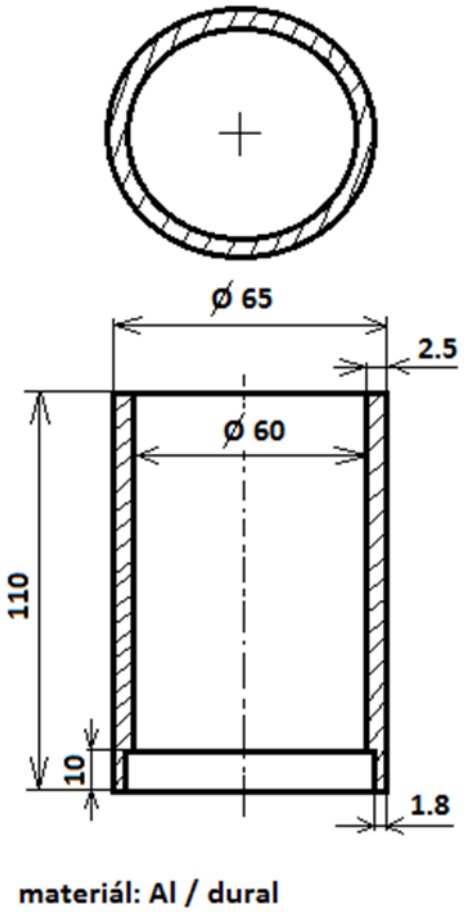
7. Mechanické provedení CanSATu

Jedna z variant mechanická konstrukce sestává ze dvou kulatách čelíček, navzájem spojených čtyřmi svorníky M3 a krytu vyrobeného z duralové trubky o vnitřním průměru 60 mm. Svorníky současně prochází čtyřmi otvory ve všech deskách PCB. Vše je zřejmé z následujících obrázků a výkresu krytu. Z něj je také zřejmé, že průměr horního čelíčka je o něco méně než 60 mm, tak aby mohl volně procházet vnitřkem krytu (trubky). Průměr dolního čelíčka je naopak větší než 60 mm, aby nemohl procházet vnitřkem krytu, ale naopak aby se zachytil v dolním zářezu. Tato konstrukce je předběžná. Je zbytečně těžká a bude potřeba je odlehčit zmenšením vnějšího průměru na cca 62.5 mm, ovšem jen ve střední části krytu. Cca 10 mm od horního okraje a 15 mm od dolního bych ponechal původních 65mm.


PragSAT


PragSAT


PragSAT

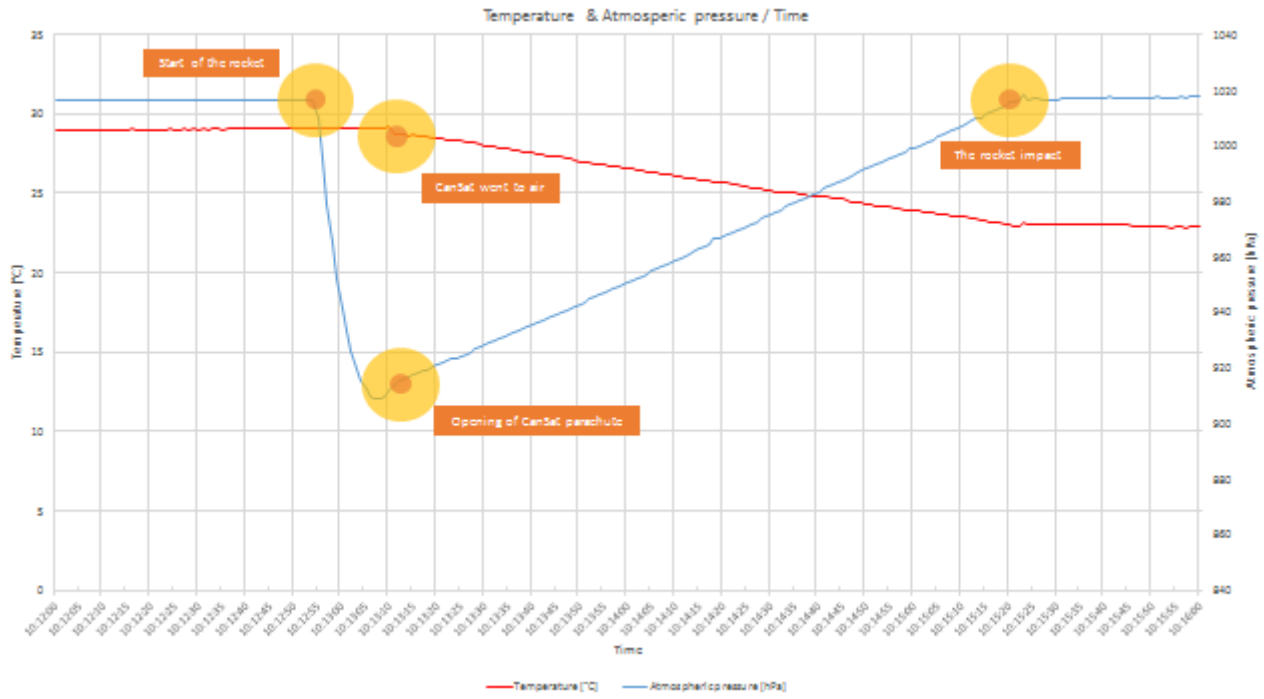


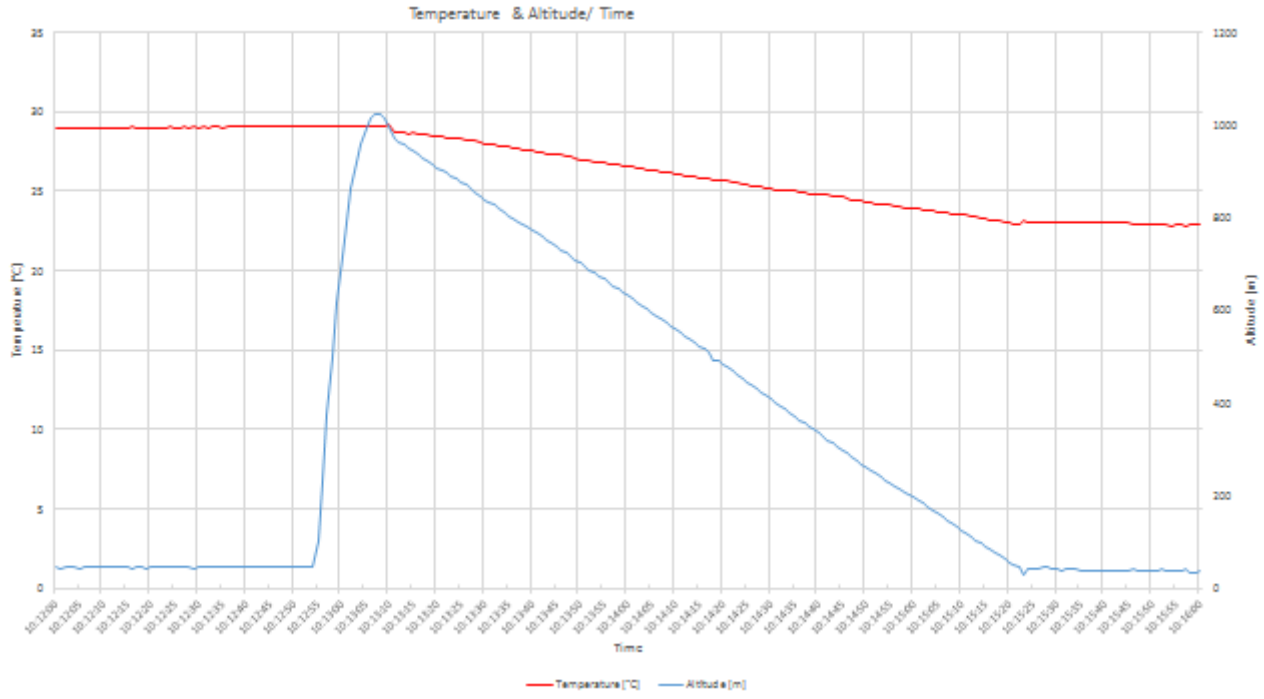
materiál: Al / dural

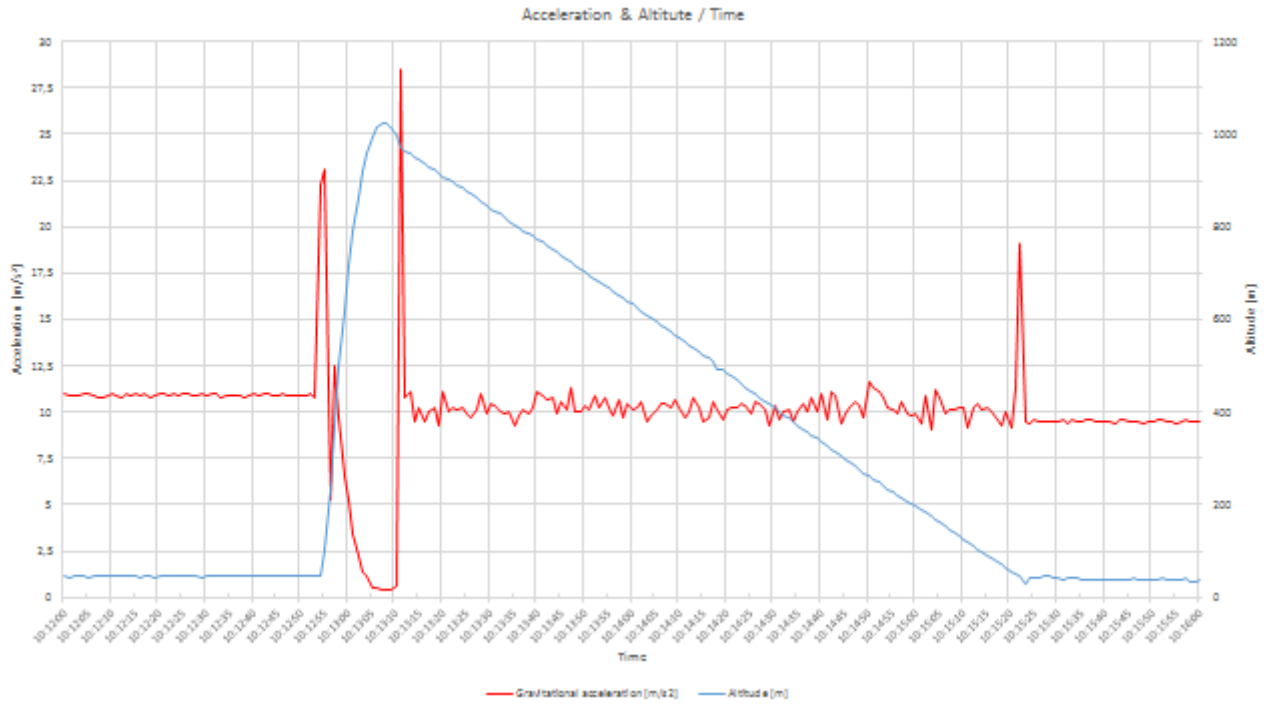
Odlehčení krytu (a tím zabezpečení váhy CanSATu vyhovující podmínkám soutěže) a současně vytvoření větrání lze docílit vyvrtáním otvorů cca 10mm v plášti. Zabezpečuje se tím i to, že teplota a tlak uvnitř CanSATu je stejná, jako v jeho okolí.

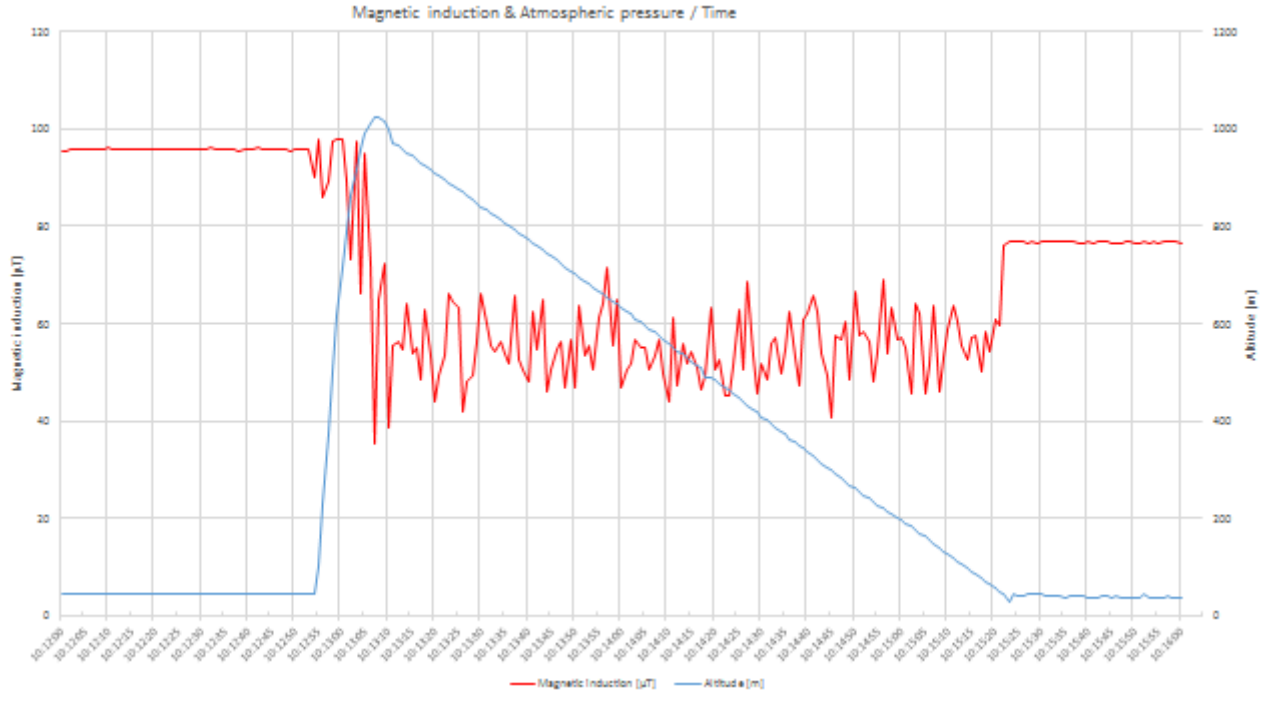
8. Zkušenosti s provozem CanSATu

Během mezinárodní soutěže ESA European Cansat Competition 1 - 5.června 2014 na základně Andøya Rocket Range v Norsku pracoval CanSAT naprosto spolehlivě. Po celou dobu vysílal naměřená data, která byla úspěšně on line zachycována pozemní stanicí a zobrazována na PC. Současně zapisoval data na SD kartu. Dále uvádím grafy z těchto dat vypracovaná studenty týmu PragSAT.


Temperature & Atmosperic pressure / Time
Start of the rocket
CanSat went to air
Opening of CanSat parachute
The rocket impact
Temperature [°C]
Atmospheric pressure [hPa]
Time
Temperature [°C]
Atmospheric pressure [hPa]


Temperature & Altitude/ Time
Temperature [°C]
Altitude [m]
Time
Temperature [°C]
Altitude [m]





9. Závěr

Věřím, že předkládaná skripta, spolu s předchozími díly z minulých let pomůžou i v budoucích letech se stavbou CanSATů a s jejich soutěžemi, ale i v dalších projektech, kde jsou potřeba mikrořadiče, čidla, přenos dat apod. Spolu s týmovou prací na těchto projektech včetně mezinárodní spolupráce to bude jistě dobrá zkušenost pro jejich budoucí firemní praxi.

10. Zdroje

[1] Váňa Vladimír: Cansat – 1.díl , , Praha 2011
[2] Váňa Vladimír: Cansat – 2.díl, programování palubního počítače s STM32, , Praha 2011

[3] Váňa Vladimír: Cansat – 3.díl , Praha 2012
[4] Váňa Vladimír: Co je to CanSAT, Praktická Elektronika – PE AR 11/2013, str.15
[5] Váňa Vladimír: Mikrokontroléry STM32F prakticky – PE AR 12/2013, str. 15
[4] Váňa Vladimír: Mikrokontroléry STM32 bezdrátově – PE AR 4/2014, str.21
[5] Wang Torstein: CanSat Competition, <http://www.rocketrange.no>
[6] WWW SPŠE Ječná k CanSatu: <http://www.spsejecna.net/cansat/>
[7] <http://www.pratthobbies.com/proddetail.asp?prod=CANSAT-1>
[8] <http://kraksat.pl/2013/01/23/map2d/>
[9] <http://aurorateam.pl/>
[10] <http://www.control.aau.dk/~jdn/edu/cansat-kit/>