VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

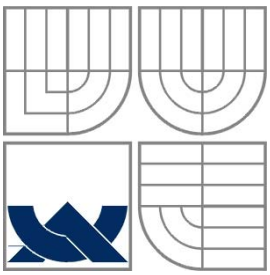
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

NÁSTROJ PRO TESTOVÁNÍ KOMUNIKACE VLOŽENÉHO ZAŘÍZENÍ POMOCÍ AT PŘÍKAZŮ

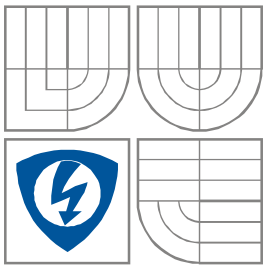
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. ALEŠ KONDR
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV RADIOELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF RADIO ELECTRONICS

NÁSTROJ PRO TESTOVÁNÍ KOMUNIKACE VLOŽENÉHO ZAŘÍZENÍ POMOCÍ AT PŘÍKAZŮ

SOFTWARE TESTER OF EMBEDDED MODULE COMMUNICATION BASED ON AT COMMANDS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Aleš Kondr
AUTHOR
VEDOUCÍ PRÁCE Ing. Michal Kubíček
SUPERVISOR

BRNO, 2008

Originál zadání

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:	Bc. Aleš Kondr
Bytem:	Na bítýškách 648 Veverská Bítýška 66471
Narozen/a (datum a místo):	21 září 1980 v Brně

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 53, Brno, 602 00
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Dr. Ing. Zbyněk Raida, předseda rady oboru Elektronika a sdělovací technika
(dále jen „nabyvatel“)

Čl. 1

Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):

   ☐ disertační práce
   ☒ diplomová práce
   ☐ bakalářská práce
   ☐ jiná práce, jejíž druh je specifikován jako .....
   (dále jen VŠKP nebo dílo)

Název VŠKP:	Nástroj pro testování komunikace vloženého zařízení pomocí AT příkazů
Vedoucí/ školitel VŠKP:	Ing. Michal Kubíček
Ústav:	Ústav radioelektroniky
Datum obhajoby VŠKP:	_____

   VŠKP odevzdal autor nabyvateli*:

   ☒ v tištěné formě – počet exemplářů: 2
   ☒ v elektronické formě – počet exemplářů: 2

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.

3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

* hodící se zaškrtněte

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.

2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.

3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti

   - ☒ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)

4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.

2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.

3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.

4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: 30. května 2008

…….. ……

Nabyvatel Autor

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta elektrotechniky a komunikačních technologií
Ústav radioelektroniky

Nástroj pro testování komunikace vloženého zařízení pomocí AT příkazů

diplomová práce

Studijní obor: Elektronika a sdělovací technika
Autor: Bc. Aleš Kondr
Vedoucí diplomové práce: Ing. Michal Kubíček

ABSTRAKT

Diplomová práce se zabývá AT příkazy a jejich použitím pro moduly bezdrátové komunikace. Tyto příkazy slouží především pro získávání a nastavování různých parametrů těchto modulů. Je zde uveden tvar AT příkazu a způsob kterým jsou do modulů odesílány. Pro vybrané bezdrátové moduly je zde soupis AT příkazů a stručný popis jejich funkce. Dále je zde uveden způsob vytváření makro bloků pro program Tera Term, které dokáží s připojeným modulem navázat komunikaci a pomocí nichž lze na komunikačních zařízeních provádět měření a testování. Toto je vhodné především pro zautomatizování tohoto procesu. Protože tento systém (program) není příliš rozšířen a znám, je zde především jeho důkladný popis, doplněný popisem měřících a testovacích makro bloků. Poslední část mé práce se věnuje vytvoření aplikace, která umožní uživateli rychle a snadno řídit a ovládat vložené zařízení. Tato aplikace je zde důkladně popsána. Z důvodů absence dokonalejšího grafického rozhraní, již není tato aplikace tvořena systémem makro bloků pro program Tera Term, ale je vytvořena v programovacím jazyku JAVA.

KLÍČOVÁ SLOVA

AT příkaz, WiFi, ZigBee, TeraTerm, JAVA, NetBeansIDE, GUI, TTL macro, komunikace po sériové lince.

BRNO UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering and Communication
Institute of Radio Electronics

Software tester of embedded module communication based on AT commands

Diploma Thesis

Study Specialization: Electronics and Communication
Author: Bc. Aleš Kondr
Supervisor: Ing. Michal Kubíček

ABSTRACT

The graduation thesis deals with AT commands and the implementation of these in modules of wireless communication. These commands are mainly used for the obtaining and setting of different parameters of such modules. A form of the AT commands and a manner in which these are sent to modules are described. For selected wireless modules, there is a set of AT commands determined together with a brief description of the functions of these. A manner of creation of macroblocks for the TeraTerm program is also mentioned; macroblocks are able to establish communication with the module connected and allow for measurement and testing of communication appliances. This is suitable especially for the automation of such a process. As such a system (program) is not widely known, a thorough description of this system is given together with a description of measuring and testing macroblocks. The last part of the thesis deals with the creation of an application enabling the user to control quickly and easily an inserted appliance. A very detailed description of this application is given. Due to an absence of an advanced graphic interface, such an application has not been created by means of a system of macroblocks for the TeraTerm program, but it has been created in the JAVA programming language.

KEY WORDS

AT command, WiFi, ZigBee, TeraTerm, JAVA, NetBeansIDE, GUI, TTL macro, communication after seriál line.

Bibliografická citace:

KONDR, A. *Nástroj pro testování komunikace vloženého zařízení pomocí AT příkazů.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2008. 54 s. Vedoucí diplomové práce Ing. Michal Kubíček.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Nástroj pro testování komunikace vloženého zařízení pomocí AT příkazů jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 30. května 2008 .....
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Michalu Kubíčkovi. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne 30. května 2008 .....
podpis autora

Obsah:

ÚVOD

Firma Beta Control s.r.o. používá pro komunikaci s vestavěnými zařízeními ZigBee moduly Xbee a Xbee – PRO firmy MaxStream a WiFi moduly OWSPA311g firmy connectBlue. Tyto moduly používají pro řízení, nastavení a čtení stavu sadu AT příkazů protokolu V.250. Cílem diplomové práce je vytvořit nástroj (program) který usnadní práci s AT příkazy. Nástroj bude zajišťovat proces měření a testování komunikačních zařízení a umožní automatizaci tohoto procesu. K tomuto nástroji má být vytvořen podrobný popis jeho funkce, který usnadní jeho případnou pozdější modifikaci a manuál umožňující snadné a rychlé seznámení se s nástrojem. Takovýto manuál by umožnil případnému dalšímu uživateli rychle a snadno proniknout do způsobu práce s tímto nástrojem a usnadnil by mu vytváření dalších modifikací pro komunikaci se zařízením pomocí AT příkazů. Požadavkem firmy BetaControl bylo i vytvoření seznamu AT příkazů včetně stručného popisu jejich funkce. Dalším krokem má být navržení metody vhodné pro rychlé a snadné řízení vložených zařízení s moduly firem MaxStream a connectBlue pomocí AT příkazu. Takováto metoda by měla jednoduchým a přehledným způsobem vytvořit dávku nastavovacích AT příkazů a pomocí takto vytvořené dávky nastavit parametry modulu do námi požadovaného stavu.

1. Moduly pracující z AT příkazy

1. 1. Zig bee moduly MaxStream X-Bee™ & X-BeePro™

Zig Bee[2] jsou bezdrátová zařízení, pracující v kmitočtovém pásmu 2,4 GHz, používající protokol přenosu 802.15.4. Většinou jde o malé kompaktní moduly s malým výkonem a malým dosahem. Moduly jsou určeny do aplikací, které vyžadují rychlostně nenáročnou bezdrátovou komunikaci řádově v desítkách Kb/s na vzdálenost několika desítek až stovek metrů. Zároveň je kladen důraz na velmi nízký příkon modulů, aby je bylo možné dlouhodobě provozovat v bateriových aplikacích. Základní parametry vzájemně pinově kompatibilních modulů jsou shrnuty v tabulce[3]: Tab. 1.

	XBee™	**XBee-PRO™**
Frekvence	ISM 2,4GHz	
Dosah – zástavba	do 30m*	do 100m*
Dosah - přímá viditelnost	do 100m*	do 1.6km*
Vysílaný výkon	1mW (0 dBm)	60mW (18dBm)** 100mW EIRP**
Přenosová rychlost RF	250kbps	
Přenos. metoda/modulace	DSSS/QPSK	
Citlivost	-92dBm	-100dBm
Napájecí napětí	2.8 – 3.4V	
Typický proud (při 3.3V)		
TX	45mA	215mA (při 18dBm)
RX	50mA	55mA
hibernate	<10uA (při 3.0V)	
sleep	<50uA (při 3.0V)	
pracovní režimy	pohotovostní, vysílácí, příjímací a konfigurační	
režimy přensu dat	transparetní a API mód	
podporované topologie síťí	point-to-point, point-to-multipoint, peer-to-peer, mesh	
počet přep. kanálů	16	12
počet uzlů	65535/ kanál	
rozměry	(2.438cm x 2.761cm)	(2.438cm x 3.294cm)
teplotní prac. rozsah	-40°C až 85°C	
typ antény	SMD, prutová, U.FL konektor viz. Obr. 1.	
certifikace	ETSI, FCC, IC	

* orientační hodnoty

** v ČR je vysílaný výkon omezen dle GL ČTÚ

Tab. 1. Parametry modulů MaxStream.

Obr. 1. Provedení antény u modulů.

Konfigurace modulů může být provedena dvěma způsoby:

- přes konfigurační mód pomocí AT příkazů
- přes API

Adresování:

Pro adresaci jednotlivých zařízení lze zvolit mezi 16 nebo 64 bitovým adresováním. U modulů MaxStream je v ZigBee síti vždy vyžadováno 64 bitové adresování, kde adresou každého modulu je jeho unikátní sériové číslo. Samozřejmostí je podpora adresování typu Unicast i Broadcast.

Komunikace:

Mezi moduly probíhá transparentně nebo v API módu a může být kryptována (128-bit AES)

- Transparetní mód – všechna data přijatá na vstupu RXD vysílacího modulu jsou transparentně přenesena na výstup TXD přijímacího modulu (náhrada sériové linky).
- API mód – data určená k bezdrátovému přenosu jsou zapouzdřena do paketů, jejichž součástí jsou konfigurační informace, není tudíž nutné nastavovat vlastnosti modulu přes konfigurační mód. API mód se ovládá pomocí AT příkazu AP.

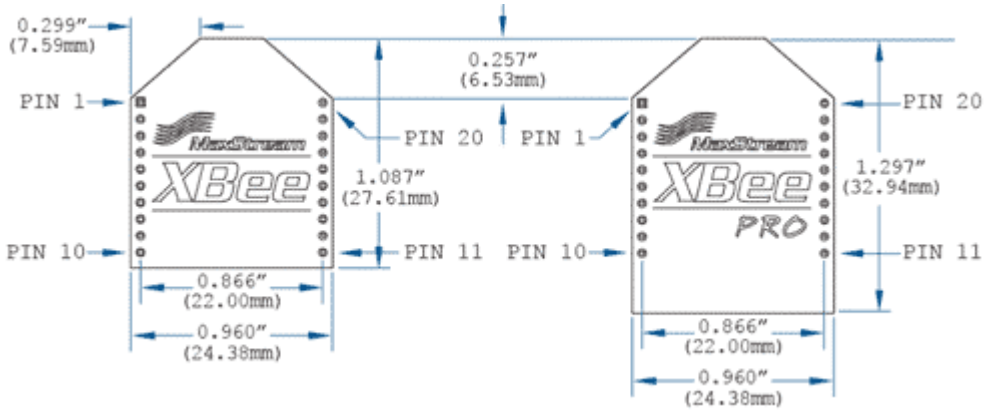
Vstupy/Výstupy:

- Sériový port RXD, TXD.
- Až 8 digitálních vstupů/výstupů.
- Až 7 analogových vstupů.
- 2 PWM výstupy.

AD převodník:

V modulech je integrován 10 bitový AD převodník s max. vzorkovací rychlostí 1kHz (rychlost je sdílená mezi aktivní analogové vstupy). Navzorkovaná data lze ukládat do zásobníku (max. 93Byte), z kterého mohou být „hromadně“ odeslána. Pokud je na straně přijímacího modulu vyžadován zpětný převod na analogový signál, je možné využít 2 PWM výstupů (DA převodník není v modulech integrován). PWM výstupy lze také využít k odečítání informace o úrovni RF signálu.

Rozměry modulů:



Obr. 2. Rozměry modulů.

Program X-CTU dodávaný výrobcem k modulům:

Pro ovládání modulů pomocí AT příkazů[4] lze použít mnoho programů. Nejjednodušší je použít program Hyperterminal, který je součástí každé instalace operačního systému Windows. Pro moduly Xbee je přímo výrobcem poskytován program X-CTU který je možno zdarma stáhnout na webových stánkách výrobce Maxstream. Tento program se skládá ze základních čtyřech oken:

PC Settings – Nastavení PC sériového portu pro připojení modulu XBee

Range Test – Testování přenosu paketů modulu XBee a sledování síly signálu. K tomuto procesu jsou potřeba moduly dva, vzájemně na sebe bezdrátově navázané a modul na druhé straně sítě musí mít propojen pin RXD a TXD.

Terminal – Nastavení a čtení parametrů modulu za použití AT příkazů. Pro vstup do režimu AT příkazů je nutno zadat "+++", po zadání trojice těchto znaků a přijetí "OK" je modul ve stavu kdy přijímá AT příkazy.

Modem Configuration - Nastavení a čtení parametrů modulu v nabídkovém menu.

Nastavení parametrů modulů je však v tomto programu nepřehledné a nekomfortní, proto je pro naše použití nedostačující.

1. 2. WiFi moduly connectBlue OWSPA311g

OWSPA311g Obr. 3. je malý bezdrátový LAN modul, zakládající se na čipu Phillips BGW211. WLAN modul byl vyvinut pro snadnou integraci do průmyslových zařízení s důrazem na velmi nízké proudové nároky. Použití tohoto modulu minimalizuje práci potřebnou k implementaci WLAN 802.11b/g, neboť poskytuje veškerý software, hardware, potřebná povolení, EMC certifikaci apod.

Obr. 3. Provedení modulu OWSPA 311g.

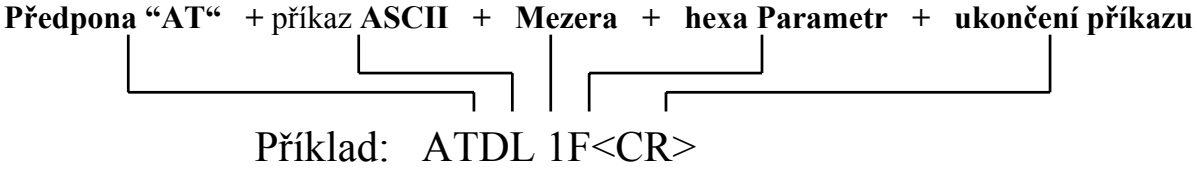
	OWSPA311g
Wireless Standard	Wireless LAN (WiFi)
LAN Verze	802.11b+g (54 Mbit/s)
Frekvence	ISM 2,4GHz
počet přep. kanálů	13
Citlivost	-82dBm
Anténa	interní/externí
Výstupní výkon	20 dBm
Dosah Int./Ext.	450/450 m
Diversita	Dual antenna diversity
Napájecí napětí	3,3 - 5,5 V
Spotřeba (min.)	0,4 mA @ 3,3V
Spotřeba (průměr Tx)	143 mA @ 3,3V
Teplotní rozsah	-30°C až +85°C
Rozměry	23 x 36 x 3,1 mm
Zabezpečení	WEP64, WEP128, WPA-PSK, WPA2-PSK,TKIP, CCMP (AES),LEAP
Certifikace	EN 60950-1:2001 and/or IEC 60950-1:2001 (1st Edition)

Tab. 2. Parametry modulu connectBlue.

2. AT příkazy

2. 1. Systém zadávání AT příkazů

AT příkazy jsou krátké textové příkazy, které se zadávají přímo v textovém módu. Každý AT příkaz musí začínat znaky AT. Dále následuje samotný příkaz, který obsahuje jeden a více znaků. Po tomto příkazu může následovat mezera, která se ale používat nemusí. Za ní se udává parametr příkazu – tj. číselná hodnota, která příkaz doprovází. Tato hodnota se zadává v hexadecimálním tvaru. Konec příkazu tvoří znak <CR>, což je vlastně značka odeslání příkazu. Celý příkaz se odesílá po sériové lince jako sada ASCII znaků. Pokud tedy odešleme například příkaz: ATCT AA1F<CR> (nastavení timeot módu), tak tento bude odeslán po sériové lince jako posloupnost číslic: 65,84,67,84,65,65,49,70,13 (číslice v dekadické soustavě). Pokud je tento příkaz v koncovém zařízení akceptován a vykonán, tak koncové zařízení odešle odpověď (acknowledgement) ve tvaru: OK<CR> což odpovídá posloupnosti: 79,75,13. Tato čísla odpovídají znakům v ASCII tabulce kterou nalezneme v příloze.



Obr. 4. Složení AT příkazu

2. 2. AT příkazy pro zařízení X-Bee a X-BeePro

Uvádím seznam příkazů standardu V.250[3], zaměřených na bezdrátovou komunikaci modulů Xbee a Xbee-PRO. Moduly vyžadují jako parametr příkazu numerické hodnoty v hexadecimálním tvaru, tzn. že se hodnota za příkazem udává ve tvaru řetězce znaků, ve kterém mohou být znaky 0 – 9 a,b,c,d,e,f. Pokud je příkaz v pořádku přijat modulem, tak je po zpětné lince vrácena sekvence znaků "OK". Pro vsup modulu do módu příjmu AT příkazu se musí zadát takzvaná escapeseqence která se skládá ze znaků: "+++" Po jejím přijetí je vrácena sekvence znaků "OK". Příkazy jsou rozděleny do následujících příkazových kategoriích:

2. 2. 1. Specielní

ATWR *Write* – Při zadání tohoto příkazu se nastavení modulu uloží do paměti EEprom.

ATRE *Restore defaults* – Nastaví parametry modulu na implicitní, tovární hodnotu.

ATFR *Software Reset* – Po přijmutí tohoto příkazu a obdržení zpětné zprávy OK se po 100ms provede reset modulu.

2. 2. 2. Síť a zabespečení

ATCH *Channel* – Tímto příkazem se dá přečíst nebo nastavit číslo kanálu, používaného pro bezdrátovou komunikaci modulu ve standardu 802.15.4. Nastavuje se hodnota 0x0B – 0x1A pro modul XBee a hodnota 0x0C – 0x17 pro modul XBee-PRO. Pokud chceme dostat číslo kanálu musíme tuto hodnotu převést do dekadického tvaru a odečíst od ní hodnotu 11. Kanály mají odstup 5 MHz a začínají na kmitočtu 2405 MHz.

ATID *PAN ID* – Tímto příkazem můžeme nastavit nebo přečíst identifikaci PAN (Personal Area Netvork). Hodnota se zadává v rozsahu 0x0000 – 0xFFFF. Pokud zadáme hodnotu na 0xFFFF tak modul nastavíme na broadcast.

ATDH *Destination Address High* – Příkaz slouží pro čtení nebo nastavení horních 32 ze 64 bitů cílové adresy, následující hodnota se zadává v rozsahu 0x0 – 0xFFFFFFFF.

ATDL *Destination Address Low* - Příkaz slouží pro čtení nebo nastavení spodních 32 ze 64 bitů cílové adresy, následující hodnota se zadává v rozsahu 0x0 – 0xFFFFFFFF.

ATMY *16-bit Source Address* – Nastaveni nebo čtení 16 bitové adresy modulu, hodnota se zadává v rozsahu 0x0000 - 0xFFFF.

ATSH *Serial Number High* – Pouze čtení horních 32 bitů sériového čísla.

ATSL *Serial Number Low* – Pouze čtení dolních 32 bitů sériového čísla.

ATRR *XBee Retries* – Nastavení nebo zjištění maximální hodnoty počtu zopakování odeslání paketu, hodnota se zadává v rozsahu 0x0 - 0x6.

ATRN *Random Delay Slots* - Nastavení nebo zjištění exponentu minimální hodnoty zpětného CSMA-CA algoritmu, který je použit pro vyhnutí se kolizi, hodnota se zadává v rozsahu 0x0 - 0x3. Pokud je hodnota nastavena na 0, vyhnutí se kolizi je vypnuto.

ATMM *MAC Mode* – Nastavení nebo zjištění hodnoty MAC Mode, hodnota se zadává v rozsahu 0x0 – 0x3 . Pokud je zadaná 0 je povolen MaxStream mód, při kterém je do paketu ukládána specielní MaxStream hlavička, pokud je zadaná hodnota 1 moduly používají paket s protokolem 802.15.4, pokud je zadaná hodnota 2 moduly požívají protokol 802.15.4 s ACK.

ATNI *Node Identifier* – Nastavení identifikační jméno uzlu. Za příkaz se vkládá se max 20 ASCII znaků, pokud je vloženo více znaků použije se jen prvních 20.

ATND *Node Discover* – Tento příkaz provede vyhledání všech modulů na stejném kanále se stejnou PAN ID adresou, vypíše jejich 16 ti bitovou adresu, sériové číslo, sílu signálu, identifikaci uzlu.

ATNT *Node Discover Time* – Nastavení nebo zjištění času jak dlouho má příkaz ND vyhledávat moduly, hodnota se zadává v rozsahu hexa 0x01 – 0xFC a je to hodnota času v hexa tvaru x100ms.

ATDN *Destination Node* - Používá se pro vyhledání nějakého NI identifikačního jména uzlu. Za příkaz se vkládá se max 20 ASCII znaků.

ATCE *Coordinator Enable* – Nastavení režimu modulu, hodnota se zadává v rozsahu 0x0 - 0x1. Při hodnotě 0x0 je modul nastaven jako koncové zařízení, při hodnotě 0x1 jako coordinator.

ATSC *Scan Channels* – Nastaví které kanály mají být skenovány, hodnota se zadává v rozsahu 0x0000 - 0xFFFF. Jde o 16ti bitové číslo, kde každému bitu nastavenému do log 1 odpovídá skenovaný kanál.

ATSD *Scan Duration* – Nastavuje se doba skenování jednotlivých kanálů, doba se nastavuje exponenciálně podle vstahu (2 ^ SD) * 15.36ms. Hodnota se zadává v rozsahu 0x00 - 0x0F.

ATA1 *End Device Association* - Nastavení parametrů modulu pokud funguje jako koncové zařízení - End Device.

ATA2 *Coordinator Association* - Nastavení parametrů modulu pokud funguje jako Coordinator.

ATAI *Association Indication* – Čtení chyb vzniklých při posledním asociování modulu.

ATFP *Force Poll* – Dotaz na nepřímou zprávu.

ATAS *Active Scan* – Délka skenování aktivního kanálů.
ATED *Energy Scan* – Délka skenování všech kanálů.
ATEE *AES Encryption Enable* – Nastavení zapnutí nebo vypnutí kódování.
ATKY *AES Encryption Key* – Nastavení kodovacího klíče.

2. 2. 3. RF Interfejs

ATPL *Power Level.* – Nastavení nebo zjištění vysílaného výkonu na výstupu modulu. Hodnota se zadává v rozsahu 0 – 4. Pro jednotlivé hodnoty se udávají následující výkony:
0 = -10/10dBm
1 = -6/12dBm
2 = -4/14dBm
3 = -2/16dBm
4 = 0/18dBm

ATCA *CCA Threshold* – Nastavení prahu CCA, což je hodnota minimálního výkonu (v – dBm) na přijímací anténě, při které jsou přijímány pakety. Hodnota se nastavuje v rozsahu 0 – 0x50

2. 2. 4. Sleep mód (Režim spánku)

ATSM *Sleep Mode* – Nastavení zapnutí a druh Sleep módu, hodnota se zadává v rozsahu 0 - 6. Při zadání hodnoty 0 je sleep mmód vypnut.
ATST *Time before Sleep* – Nastavení doby zapnutí Sleep módu (čas, za jak dlouho po nečinnosti sériového přenosu, nebo nečinnosti radiové komunikace bude aktivován sleep mód), hodnota se zadává v rozsahu 0x0000 - 0xFFFF a udává čas x1ms.
ATSP *Cyclic Sleep Period* – Nastavení času cyklického režimu spánku. Hodnota se zadává v rozsahu 0 – 0x68B0 a udává čas x10 ms.
ATDP *Disassociated Cyclic Sleep Period* - Nastavení času cyklického režimu spánku. Hodnota se zadává v rozsahu 0 – 0x68B0 a udává čas x10 ms.

2. 2. 5. Sériový interfejs (COM port)

ATBD *Interface Data Rate* – Nastavení rychlosti sériové linky po které komunikuje modul s hostujícím zařízením. Hodnota se zadává v rozsahu 0-7 odpovídající rychlostem:
0 = 1200bps
1 = 2400bps
2 = 4800bps
3 = 9600bps
4 = 19200bps
5 = 38400bps
6 = 57600bps
7 = 115200bps

ATRO *Packetization Timeout* – Nastavení času mezi znaky v sériovém přenosu.
ATAP *API Enable* – Nastavení zapínání API módu.
ATPR *Pull-up Resistor Enable* – Nastavení zapnutí upínacího rezistoru na log 1.

2. 2. 6. Nastavení vstupů a výstupů

ATD8 *DI8 Configuration* – Konfigurace portu DI8.

ATD7 *DIO7 Configuration* – Konfigurace portu DI7.
ATD6 *DIO6 Configuration* – Konfigurace portu DI6.
ATD5 *DIO5 Configuration* – Konfigurace portu DI.
ATD0–D4 *(DIO4 -DIO4) Configuration* - Hromadná konfigurce portu DI0 – DI4.
ATIU *I/O Output Enable* – Zapínání a vypínání vstupně výstupních portů.
ATIT *Samples before TX* – Nastavení čísla DIO a ADC vzorků.
ATIS *Force Sample* – Přečtení urovní na vstupech.
ATIC *DIO Change Detect.* – Povolení nebo zakázání monitorování změny na vstupech DIO. 0 - 7
ATIR *Sample Rate* – Nastavení vzorkovací rychlosti.
ATAV *ADC Voltage Reference* – Nastavení vnitřní nebo vnější reference.
ATIA *I/O Input Address* – Čtení nebo nastavení adresy modulu.
ATT0–T7 *(D0 - D7) Output Timeout* – Hodnota výstupního Timeout parametru.
ATP0 *PWM0 Configuration* – Nastavení výstupního portu P0.
ATP1 *PWM1 Configuration* – Nastavení výstupního portu P1.
ATM0 *PWM0 Output Level* – Nastavení výstupní úrovně PWM0 linky.
ATM1 *PWM1 Output Level* - Nastavení výstupní úrovně PWM1 linky.
ATPT *PWM Output Timeout* – Nastavení hodnoty timeout u PWM výstupu.
ATRP *RSSI PWM Timer* - Nastavení hodnoty timeout u PWM výstupu.

2. 2. 7. Diagnostika

ATVR *Firmware Version* – Po zadání tohoto příkazu se nám vrátí zpět hodnota firmware verze modulu. Vrácená hodnota je v rozsahu 0 – 0xFFFF.
ATVL *Firmware Version – Verbose* – Po zadání tohoto příkazu se nám vrátí detaily verze modulu.
ATHV *Hardware Version* – Tento příkaz nám zjišťuje verzi hardware modulu. Vrácená hodnota je v rozsahu 0 – 0xFFFF.
ATDB *Received Signal Strength* - Tento příkaz zjišťuje sílu signálu naposledy přijatého packetu. Vrácená hodnota má rozsah 0 – 0x64, což po převedení na dekadický tvar nám udává sílu signálu v -dBm
ATEC *CCA Failures* – resetování CCA počítadla.
ATEA *ACK Failures* – resetování ACK počítadla.

2. 2. 8. Nastavení AT příkazů

ATCT *Command Mode Timeout* - Nastavení času, po který když nedojde k zadání dalšího AT příkazu, tak dojde k ukončení módu AT příkazů, hodnota se zadává v rozsahu 0x0000 – 0xFFFF a udává hodnotu času x100ms.
ATCN *Exit Command Mode* – Zadání příkazu způsobí návrat z módu pro zadávání AT příkazů, do normálního módu.
ATAC *Apply Changes* – Zadání příkazu způsobí uložení změn a znovu inicializace modulu.
ATGT *Guard Times* - Nastavení času, jak dlouho se má čekat na vstup do módu AT příkazů po zadání znaků pro vstup do tohoto módu.
ATCC *Command Sequence Character* – Nastavení znaku který se má použít pro vstup do módu AT příkazů, hodnota se zadává v rozsahu 0 – 0xFF (1 byte), což odpovídá číslu znaku v ASCII tabulce.

2. 3. AT příkazy pro connectBlue

U těchto modulů se AT příkazy [5] zadávají podobným způsobem jako u modulů Xbee a Xbee-PRO, odlišné však je zadávání parametru příkazu. Ten se nezadává v hexadecimálním formátu, ale pouze v dekadickém. Escape sequence pro tento modul je: "///"

2. 3. 1. Standardní AT příkazy

AT *Attention command* – Příkaz slouží ke zjištění zdali je připojeno zařízení k COM portu a zdali je v režimu AT příkazů.

AT* *List available commands* - Po zadání tohoto příkazu nám modul vrátí zpět všecky podporované AT příkazy.

AT&F *Restore to factory settings* - Tento příkaz nastaví všechny parametry modulu na hodnotu nastavenou v továrně.

AT&F1 *Restore to static default settings* - Nastaví modul do základního nastavení. Stejný jako předchozí příkaz

ATS2 *Escape character* - Tímto příkazem se nastavuje znak tzv. escape seqence, kterým se vstupuje do módu AT příkazů.

ATS3 *Command line termination character* - Tímto příkazem se nastavuje znak, který má být použit jako <CR>.

ATS4 *Response formatting character* - Tímto příkazem se nastavuje znak, který má být použit jako <LF>.

ATS5 *Backspace character* - Tímto příkazem se nastavuje znak, který má být použit jako <BS>.

2. 3. 2. Příkazy pro linkovou vrstvu

AT*AGAM *Authentication mode* - Nastavuje mód ověření. Možností jsou: Otevřená síť / sdíleno / automatické nastavení.

AT*AGEM *Encryption mode* - Nastavení kódovacího módu. Možnosti jsou. žádné kódování / WEP64 / WEP128 / TKIP / ASS/CCMP.

AT*AGSM *Security mode* - Nastavení zabezpečení. Možnosti jsou: bez zabespečení / sdílené WEP64 / sdílené WEP128 / WPA-PSK-TKIP / WPA2-PSK-TKIP.

AT*AGOM *Operational mode* - Nastavení operačního módu. Možnosti jsou: žádný / Infrastructure / Ad-Hooc.

AT*AGFP *Encryption/authentication key* - Tímto příkazem se zadává kódovací klíč.

AT*AGFPRI *Read encryption/authentication key* - Tímto příkazem můžeme pouze zjistit, získat kódovací klíč patřičného indexu.

AT*AGFPWI *Write encryption/authentication key (with index)* - Tímto příkazem můžeme na zadat kódovací klíč na určitý index(1 - 4).

AT*AGAFP *Active encryption/authentication key* - Příkaz slouží na aktivaci kódovacího klíče pod určitým indexem (1 - 4).

AT*AGSSID *ESSID* - Tímto příkazem se nastavuje identifikátor bezdrátové sítě access pointu. Zadávat se může max 32 asci znaků.

AT*GRSS *RSSI Value* - Příkaz slouží na zjištění síly přijímaného signálu. Vrácená hodnota 128 odpovídá síle signálu 0dBm.

AT*AGCH *Channel number* - Příkaz slouží na nastavení kanálu, který budeme používat pro komunikaci. Zadána hodnota 0 znamená automatickou detekci čísla kanálu.

AT*AGRTE *Data rate and link adaptation* - Příkaz slouží na nastavení přenosové rychlosti v rozsahu 1Mbit – 54Mbit. Také zde můžeme nastavit automatickou volbu přenosové rychlosti.

2. 3. 3. Příkazy pro síťovou vrstvu

AT*ANIP *IP address* - Tímto příkazem se nastavuje IP adresa modulu maska podsítě a výchozí brána.

AT*ANDHCP *DHCP activation* - Příkaz slouží pro aktivaci DHCP dynamické přidělování IP adresy.

AT*ANHN *Hostname* – Nastavení jména hostname použínaného s dynamickou DNS.

2. 3. 4. Příkazy datového módu

AT*ADDM *Enter data mode* - Návrat z módu zadávání AT příkazů zpět do datového módu.

AT*ADMRP *Read maximum number of remote peers* – zjištění maximálního počtu dálkových přístupů.

AT*ADNRP *Number of remote peers* – Číslo upřednostňovaného dálkového přístupu.

AT*ADRDRP *Read default peer* – Zjištění parametrů základního dálkového přístupu.

AT*ADWDRP *Write remote peer information* - Nastavení parametrů základního dálkového přístupu.

2. 3. 5. Informační příkazy

AT*AILBA *Read MAC-address* – Zjištění MAC adresy modulu.

AT*AILVI *Reads local version information* – Zjištění informací o modulu.

2. 3. 6. Ostatní příkazy

AT*AMRS *RS-232 setting* - Příkaz slouží pro nastavení parametrů sériové linky modulu. Nastavují se zde parametry jako rychlost, počet datových bitů, počet stop bitů, parita, kontrola toku.

AT*AMET *Escape sequence timing setting* – Nastavení časů před a po zadání escape sequence.

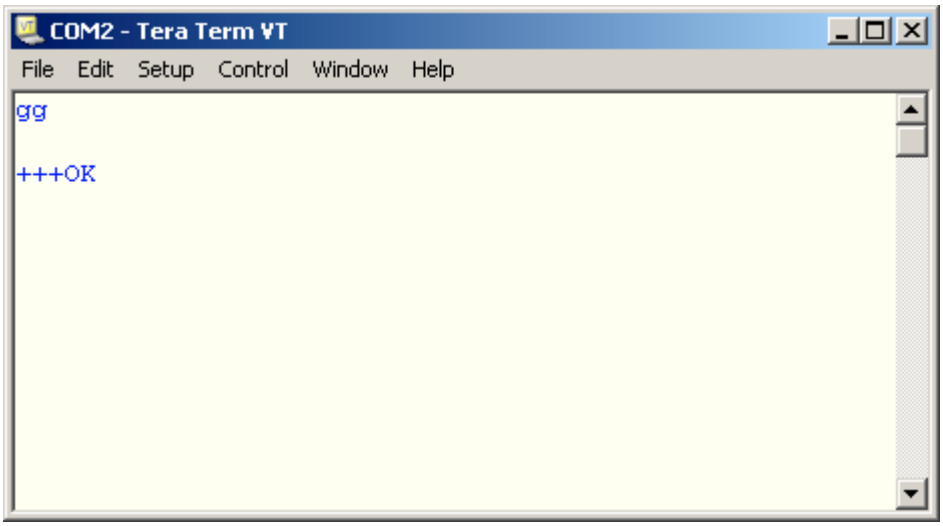
AT*AMWS *Watchdog settings* - Příkaz slouží pro restartování modulu.

AT*AMPM *Power mode* – Nastavení módu z hlediska spotřeby modulu.

3. Měření a testování v programu Tera Term

3. 1. Program Tera Term

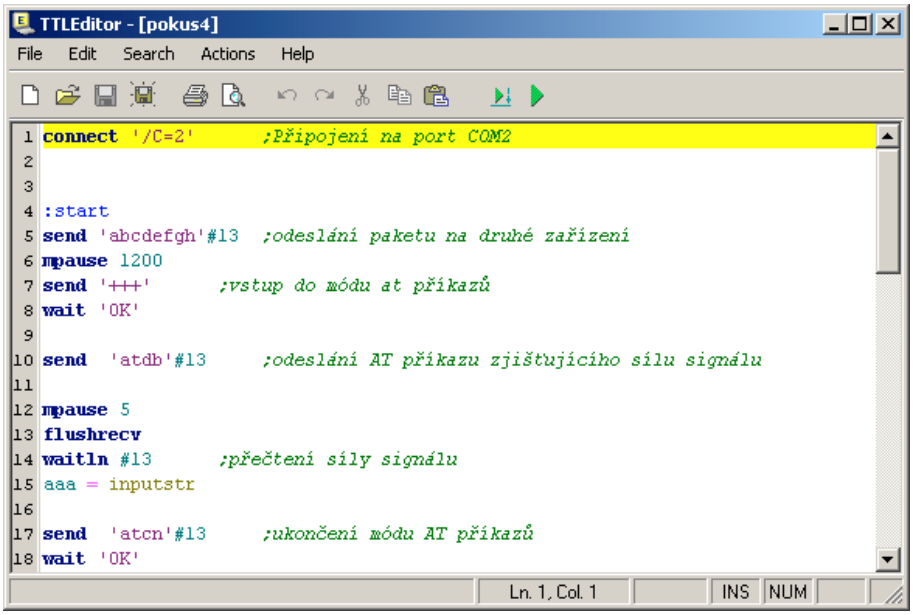
Program Tera Term[6] je terminálový program podobný nástroji Hyperterminal, který je standardní součástí operačního systému Windows. Program TeraTerm je součástí instalačního balíčku volně dostupného na internetu, který obsahuje i jiné užitečné programy, které však k řešení projektu nejsou využity. Pomocí programu TeraTerm se můžeme připojit k sériovému portu, posílat a příjmat z něj data. Vysílaná a přijímaná data se zobrazují v okně programu jako ASCII znaky. Při stisku klávesy je okamžitě odeslán ASCII kód příslušného znaku na sériovou linku. Balíček Tera Term obsahuje také program: ttpmacro. Je to program který dokáže zpracovávat MACRO soubory s příponou *.ttl, dokáže se připojit k programu Tera Term a automaticky přes něj odesílat a přijímat data z COM portu.



Obr. 5. Okno programu Tera Term.

3. 2. Jazyk MAKRO Tera Term (TTL)

Programy pro makro se píší v klasickém textovém módu do souboru s označením *.ttl. Makro soubory můžeme psát a editovat buďto v klasickém textovém editoru poznámkový blok a nebo můžeme využít programu dodávaného v balíku Tera Term, který se jmenuje: TTLEditor. Tento program je určený přímo na psaní a editaci souborů *.ttl. Hlavní předností tohoto programu je jeho barevné zvýraznění syntaxe napsaného kódu (viz Obr. 8) a možnost ladění, tedy spouštění kódu přímo z editoru, včetně možnosti zastavení jeho běhu na zvoleném řádku (breakpoint).



Obr. 6. Program TTLEditor určený na editaci makro souborů *.ttl.

3. 2. 1. Formát zápisu řádků

Řádek který nemá vliv na chod programu:

Řádek který neobsahuje žádný znak, nebo obsahuje mezeru, tabelátorový znak, nebo komentář. Komentář je řádek začínající středníkem ";", všechny znaky které jsou umístěny za tímto znakem, se až ke konci řádku se považují za komentář. Takový řádek nemá žádný vliv na běh makra.

```
Příklad:
    ; Tera Term Language
```

Řádek s příkazem:

Takový řádek obsahuje jeden příkaz s parametrem.

```
Formát:
    <příkaz> <parameter> ...

Příklad:
    connect'myhost'
    wait 'OK' 'ERROR'
    if result=2 goto error
    sendln 'cat'
    pause A*10
    end
```

Přiřazovací řádek:

Tento řádek obsahuje převodní povel "=".

```
Formát:
    <Proměnná> = <Hodnota (konstanta, proměnná, výraz)>

Příklad:
     A = 33
     B = C              C musí být vždy hodnota.
     VAL = I*(I+1)
```

```
    A=B=C               Hodnota B=C (0 pro false, 1 pro true)
                        je zapsaná do A.
    Error=0<J
    Username='MYNAME'
```

Řádek návěští:
Tento řádek začíná vždy znakem dvojtečka ':', následující znaky identifikují návěští, což je místo odkud startuje podprogram a kam se směřují skoky v programu.

```
Formát:
    :<Návěští>
Příklad:
    :dial
    :100
```

3. 2. 2. Identifikátory a reservovaná slova

Identifikátory proměnných:
První znak musí být abecední (A-Z, a-z), nebo znak podtržítka "_". Další znaky mohou být abecední, podtržené, nebo numerické (0-9). Identifikátor proměnných není citlivý na velká písmena. Maximální délka proměnné je 32 znaků.

```
Příklad:
    VARIABLE
    _flag
```

Identifikátory návěští:
Identifikátor návěští se může skládat ze znaků abecedních, podtržítka a numerických znaků a není citlivý na velká písmena. Maximální délka e 32 znaků.

```
Příklad:
    label1
    100
```

Rezervovaná slova:
Následující slova jsou rezervovaná a nedají se proto použít jako název proměnných a návěští:

```
[příkazy]
   bplusrecv, bplussend, changedir... (viz seznam příkazů)
[Operatory]
   and, not, or, xor
[Systemové proměnné]
   groupmatchstr1, groupmatchstr2, groupmatchstr3, groupmatchstr4,
   groupmatchstr5, groupmatchstr6, groupmatchstr7, groupmatchstr8,
   groupmatchstr9, inputstr, matchstr, param2, param3, result, timeout
```

3. 2. 3. Proměnné

TTL používá pouze dva druhy proměnných:
Integer:
32 bitová proměnná, která může nabývat hodnot od -2147483648 do 2147483647.
String (znaková):
Sekvence obsahující znaky, mimo NUL. Znak může nabývat hodnot 1-255. Maximální počet znaků v proměnné je 255.

3. 2. 4. Tipy proměnných

Uživatelem definované proměnné:

Definuje je uživatel. Typ proměnných se rozhoduje podle hodnoty kterou uchovává na začátku (integer nebo string). Jakmile se rozhodne o typu proměnné, tak ten se už nedá změnit.

Systémové proměnné:
Všechny systémové proměnné mají předdefinovaný typ a hodnotu. Užívají je příslušné příkazy.

Proměnná	Typ		Příkazy které ji používají
groupmatchstr1 groupmatchstr2 groupmatchstr9	string	""	waitregex
inputstr	string	""	recvln, waitln, waitrecv, waitregex, passwordbox, inputbox
matchstr	string	""	waitregex
param2	string	*1	*1
param3	string	*1	*1
result	integer	0	bplussend, bplusrecv, clipb2var, kmtfinish, kmtget, kmtrecv, kmtsend, ifdefined, quickvanrecv, quickvansend, recvln, var2clipb, wait, waitevent, waitln, waitrecv, waitregex, xmodemrecv, xmodemsend, zmodemrecv, zmodemsend, str2int, strcompare, strlen, strscan, filereadln, filesearch, filestrseek, yesnobox
timeout	integer	0	recvln, wait, waitevent, waitln, waitrecv, waitregex

*1 Druhý a třetí parametr spuštění MACRO. První parametr je jméno makra. Viz "Command line".

3. 2. 5. Konstanty

Konstanta tipu – Integer:
Typ integer se zapisuje jako dekadické číslo, nebo hexadecimal číslo které začíná znakem "$"

```
Example:        123
                -11
                $3a
                $10F
```

Poznámka k negativnim konstantám tipu integer:
Použití negativních konstant tipu integer může způsobit problém viz dále.
Pro příklad: `for i -10 0`
Způsobí syntax chybu, protože druhý parametr je považován jako "i-10" místo "i". Pokud se chceme vyhnout těmto problémům, můžeme použít následující řešení:
1) položit "0" před "-".

```
for i 0-10 0
```

2) Přidělit negativní konstantu proměnné.

```
A = -10
for i A 0
```

Konstanta tipu – String:
Jsou dva způsoby jak můžeme zadávat konstanty tipu string:
a) Řetězec znaků se zapisuje mezi znaky ' nebo " (tyto znaky musí být na obou dvou stranách stejné).

```
Příklad:        'Hello, world'
                "I can't do that"
```
b) Jeden znak můžeme zapsat jako "#" následován číslem ASCII kódu (dekadické nebo hexadecimalní číslo). Poznámka: Strings nemůže obsahovat znak NUL (ASCII code 0).
```
Příklad:        #65     Znak "A".
                #$41    Znak "A".
                #13     The CR character.
```
Formát a) a b) může být navzájem kombinován.
```
Příklad:        'cat readme.txt'#13#10
                'abc'#$0d#$0a'def'#$0d#$0a'ghi'
```

3. 2. 6. Výrazy a operátory

Výrazy se skládají z konstant, proměnných, operátorů a závorek. Konstanty a proměnné musí být tipu integer. Hodnota výrazu je také typu integer. Hodnota relačního výrazu (výraz používající větší menší rovná se) je 0 pokud je pravda(true) a 1 pokud není pravda(false).

Kategorie	Priorita	Operace
Unární	1, vysoká	not
Multiplikativní	2	* / %
Aditivní	3	+ - or xor
Relační	4, nízká	= <> < > <= >=

Poznámka: Hodnota výrazu A % B je zbytek z podílu A / B.
```
Příklad:
    1 + 1
    4 - 2 * 3   Hodnota je -2.
    15 % 10     hodnota je 5.
    3 * (A + 2) A je nějaká hodnota integer.
    A and not B
    A <= B              A a B je proměnná integer. Hodnota je 0, pokud
                        je výsledek pravda(true), nebo 1 pokud je
                        nepravda(false).
```

3. 3. Příkazy pro MAKRO - TTL

3. 3. 1. Komunikační příkazy

changedir <path> Změní základní adresář Tera Termu na: <path>.
clearscreen <int> Způsobí vymazání okna VT pokud <int> je 0. Způsobí vymazání scroll buferu a okna VT pokud <int> je 1. Způsobí vymazání obrazovky okna TEK pokud <int> je 2.
closett Provede odpojení hostujícího zařízení a zavře okno Tera Term.
connect <command line parameters> Nás bude nejvíce zajímat komunikace přes COM port, do <command line parameters> zadáme: '/C=x' kde x bude číslo COM portu.
disconnect Uzavře spojení mezi Tera Termem a hostujícím zařízením.
enablekeyb <flag> Zakáže nebo povolí vstup klávesnice do Tera Termu.
flushrecv Vymaže přijaté znaky v bafru MACRO.

gettitle <strvar> Přepíše text titulku hodnotou uloženou v textové proměnné: <strvar>.
loadkeymap <filename> Nahraje nastavení klávesnice ze souboru <filename>.
logclose Zavření logovacího souboru.
recvln Přijme od hostujícího zařízení z linky znaky a uloží je do systémové proměnné "inputstr".
restoresetup <filename> Nahraje nastavovací soubor se jménem <filename>.
send <data1> <data2> Odešle z Tera Termu znaky na hostující zařízení.
sendbreak Odešle na hostující zařízení signál přerušení.
sendfile <filename> <binary flag> Odešle soubor se jménem <filename> na hostující zařízení.

3. 3. 2. Řídící příkazy

call <label> Skok do podprogramu začínajícího návěští <label>.
end Způsobí ukončení chodu makra a MACRO uzavře.
execcmnd <statement> Vykoná příkaz který je obsažený v proměnné <statement>.
exit Návrat z *.ttl podsouboru do hlavního *.ttl souboru.
for <intvar> <first> <last> Opakuje obsah mezi "for" a "next", dokud integer proměnná <intvar> nenabyde hodnoty <last> .
goto <label> Skok programu na návěští <label> , bez návratu.
if <int> <statement> Provede příkaz <statement>, pokud <int> není nula.
ifdefined <var> Zjištění typu proměnné <var>. Tip proměnné vyhodnotí do systémové proměnné "result":
include <include file name> Spuštění podsouboru *:ttl jehož název je v <include file name>.
mpause <time> Pauza na dobu <time> v millisekundách.
pause <time> Pauza na dobu <time> v sekundách.
return Návrat z podprogramu do hlavního programu.
while <int> Opakuje část programu mezi 'while' a 'endwhile' dokud <int> není pravda.

3. 3. 3. Příkazy na přepočet mezi proměnými

code2str <strvar> <ASCII code> Pokud je hodnota integer <ASCII code> v rozsahu 1-255, tento příkaz zkopíruje znak obsažený v <ASCII code> do proměnné tipu string <strvar>.
int2str <strvar> <integer value> Převede <integer value> proměnnou tipu integer do proměnné tipu string <strvar>.
str2code <intvar> <string> Pokud <string> se skládá z jednoho znaku, tento příkaz zkopíruje ASCII kód tohoto znaku do proměnné tipu integer <intvar>.
str2int <intvar> <string> Převede číselné znaky <string> které jsou v proměnné string do jejich numerické podoby, proměnné tipu integer <intvar>.
strcompare <string1> <string2> Porovná dvě proměnné tipu string. Výsledek porovnání těchto dvou proměnných se projeví na systémové proměnné "result":
strconcat <strvar> <string> Na konec proměnné string <strvar> připojí druhou proměnnou tipu string <string>.
strcopy <string> <pos> <len> <strvar> Vykopíruje část proměnné <string> do druhé proměnné <strvar>. Vykopírovaná část začíná znakem <pos> v pořadí a má délku <len>.
strlen <string> Do systémové proměnné "result", uloží počet znaků proměnné <string>.
strscan <string> <substring> Hledá proměnou <substring> v proměnné <string>.

3. 3. 4. Příkazy pro práci se soubory

fileclose <file handle> Zavře soubor označený <file handle.
fileconcat <file1> <file2> Připojí kopii souboru <file2> na konec souboru <file1>.

filecopy <file1> <file2> Zkopíruje soubor <file1> do souboru <file2>.
filecreate <file handle> <filename> Vytvoří a otevře nový soubor s názvem <filename>.
filedelete <filename> Smaže soubor s označením <filename>.
filemarkptr <file handle> Označí ukazatel souboru na otevřený soubor nazvaný <file handle>.
fileopen <file handle> <file name> <append flag> Otevře soubor nazvaný <file name>.
filereadln <file handle> <strvar> Čte řádek ze souboru označeného <file handle>.
fileread <file handle> <read byte> <strvar> Čte určitý bajt ze souboru specifikovaného <file handle>.
filerename <file1> <file2> Přejmenuje <file1> na <file2>. <file1> a <file2> nemusí být stejné.
filesearch <filename> Vyhledá soubor pojmenovaný <filename>.
fileseek <file handle> <offset> <origin> Přesun souborového ukazatele souboru <file handle>.
fileseekback <file handle> Přesun souborového ukazatele nějakého souboru do pozice označené "filemarkptr". Soubor je specifikován <file handle>.

3. 3. 5. Příkazy pro práci z heslem

delpassword <filename> <password name> Smaže heslo specifikované <password name> v souboru pro ukládání hesel <filename>.
getpassword <filename> <password name> <strvar> Získá zakódované heslo označené <password name> ze souboru hesel <filename>.
passwordbox <message> <title> Zobrazí dialogové okno vyzívající k uložení hesla.

3. 3. 6. Ostatní příkazy

beep Vydá zvukový signál.
closesbox Zavře dialogové okno status otevřené příkazem "statusbox".
clipb2var <strvar> Zkopíruje data obsažená v paměti do proměnné tipu string <strvar>.
exec <command line> Spustí aplikaci s názvem <command line>.
getdate <strvar> Aktuální datum vloží do proměnné <strvar>, ve formátu: "YYYY-MM-DD".
getenv <envname> <strvar> Vezme hodnotu prostředí označenou jako <envname> a uloží do proměnné tipu string <strvar>.
gettime <strvar> Aktuální čas vloží do proměnné <strvar>, ve formátu: "HH:MM:SS".
inputbox <message> <title> Zobrazí dialogové okno umožňující uživateli vkládat text.
messagebox <message> <title> Zobrazí dialogové okno ze správou <message> a titulkem <title>.
random <integer variable> <max value>
Vygeneruje náhodné číslo v rozsahu 0 až <max number>-1 a uloží do proměnné <integer variable>. <max number> musí být větší než 0.
setdate <date> Nastaví systémové datum na hodnotu <date>.
setdlgpos <x> <y> Změní počáteční pozici dialogových okenzobrazovaných příkazy: "inputbox", "messagebox", "passwordbox" a "statusbox".
setexitcode <exit code> Nastavení výstupní hodnoty pro MACRO hodnota tipu integer <exit code>.
settime <time> Nastaví systémového času na hodnotu <time>. Formát <time> musí být: "HH:MM:SS".
show <show flag> Minimalizuje MACRO, pokud <show flag> je nula. Obnoví MACRO, pokud <show flag> je větší než nula. Skryje MACRO, pokud <show flag> je menší než nula.
statusbox <message> <title> Pokud již není zobrazeno, tak zobrazí dialogové okno.
var2clipb <string> Zkopíruje <strvar> do paměti.

yesnobox <message> <title> Zobrazí dialogové okno <message>, <title>, které obsahuje tlačítka: "Yes" a "No".

3. 4. Příklady maker

V následující kapitole bylo použito poznatků z předchozí kapitoly z příkazů pro makro a systému programování maker a bylo vytvořeno pár měřících a testovacích programů pro práci z AT příkazy. Tyto programy vyžadují ke své činnosti připojení modulů X-Bee na sériovou linku COM2.

3. 4. 1. Posílání AT příkazů

Makro (Výpis kódu 1.) provede připojení k sériové lince příkazem *connect*, poté pomocí příkazů *send* posílá na tento port znaky. Příkaz *wait* slouží k pozastavení chodu programu do doby, než ze sériové linky přijde požadovaná odpověď, v našem případě je to dvojice znaků OK.

```
connect '/C=2'

for i 1 10

int2str por i
statusbox por 'Kolikrát'

send  '+++'
wait 'OK'
send  'atd0 4'#13
wait 'OK'
send  'atcn'#13
wait 'OK'
beep
mpause 1000

send  '+++'
wait 'OK'
send  'atd0 5'#13
wait 'OK'
send  'atcn'#13
wait 'OK'
beep
mpause 1000

next

closett
end
```

Výpis kódu 1. Makro pro zasílání AT příkazů na COM.

3. 4. 2. Macro na čtení síly signálu

Makro (Výpis kódu 2.) se připojí k sériové lince COM2 a odešle libovolnou sekvenci znaků. Následně se odešlou znaky "+++" které způsobí vstup připojeného modulu do módu AT příkazů. Proběhne odeslání příkazu *ATDB,* který nám v odpovědi zpět vrátí v hexa tvaru hodnotu síly signálu posledně přeneseného paketu. Tato hodnota je přesunuta do proměnné *aaa,* přepočtena na dekadický tvar a zobrazena. Celý tento program neustále cikluje. Viz Obr. 10.

```
connect '/C=2'          ;Připojení na port COM2

:start
send 'abcdefgh'#13      ;odeslání nějakých dat na druhé zařízení
mpause 1200
send '+++'              ;vstup do módu at příkazů
wait 'OK'

send  'atdb'#13         ;odeslání AT příkazu zjišťujícího sílu signálu
mpause 5
flushrecv
waitln #13              ;přečtení síly signálu
aaa = inputstr
send  'atcn'#13         ;ukončení módu AT příkazů
wait 'OK'

HtD_in = aaa
call hextodec   ;Zavolání podprogramu na přepočet hexa string na integer

int2str vy2 HtD_out
zobr = '- '
strconcat zobr vy2
strconcat zobr ' dB'
statusbox zobr 'Síla signálu:'  ;Zobrazení síly signálu

goto start
```

Výpis kódu 2. Makro pro měření síly signálu.

3. 4. 3. Podprogram pro převod proměnných

Pokud do proměnné *HtD_in* zapíšeme proměnnou tipu *string* v hexadecimálním tvaru a zavoláme příkazem *call hextodec* tento podprogram, tak se nám po návratu vrátí v proměnné *HtD_out* původní hodnota v proměnné *integer*.

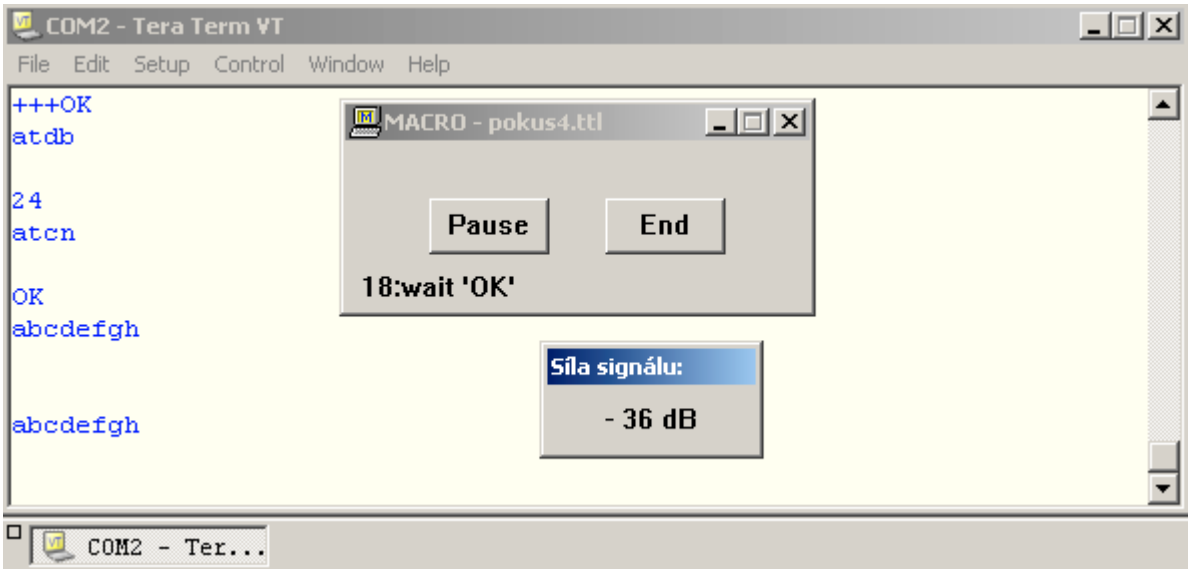
```
:hextodec       ;převod "HtD_in" = String hexa do "HtD_out" = Integer dekad
 strlen HtD_in
 HtD_b = result
 HtD_g = 1
 HtD_out = 0

 for HtD_c HtD_b 1
  strcopy HtD_in HtD_c 1 HtD_d
  str2int HtD_e HtD_d

  if result=0 then
   str2code HtD_f HtD_d
   HtD_e = HtD_f - 55
  endif

  HtD_h = HtD_g * HtD_e
  HtD_g = HtD_g * 16
  HtD_out = HtD_out + HtD_h
 next
return
```

Výpis kódu 3. Makro pro převod ze string hexa do integer.



Obr. 7. Obrazovka spuštěného makra.

4. Rychlé a snadné řízení modulů pomocí AT v prostředí JAVA

4. 1. JAVA

Java je objektově orientovaný programovací jazyk, rozsáhlá počítačová technologie a počítačová platforma. Tento jazyk byl vyvinutý firmou Sun Microsystems a syntaxí vychází z jazyka C++. Oproti C++ však Java neobsahuje žádné složité konstrukce (jako jsou např. ukazatele), které jsou poměrně komplikované na použití a kladou tak velké nároky na programátora.

Díky automatické správě paměti (garbage collector), se programátor nemusí zabývat o manuální čištění paměti po nepotřebných objektech. Nahrazením ukazatelů (pointers) odkazmi (references) je ukončená hrozba zápisu do neplatné paměti. Mechanizmus vláken umožňuje tvorbu propracovaného grafického uživatelského rozhraní (GUI). Navíc Java obsahuje serializaci, která přináší elegantní řešení v případě, že je potřeba ukládání dát do souborů anebo jejich přenášení po síti. Standardně dodávané knihovny umožňují bez dalších doplňků okamžitě vytvářet uživatelské rozhraní, pracovat ze soubory, textem, databázemi, komprimovanými soubory, předvolbami a dalšími částmi počítačového prostředí.

Programy napsané v Javě jsou přenositelné na každou platformu (ať už Windows, Unix, Linux anebo Solaris), na níž je nainstalován a spuštěn virtuální stroj jazyka Java (Java Virtual Machine – JVM). Kód je kompilovaný jen jednou na tzv. bajtový kód (byte code), který je při každém spuštění interpretovaný virtuálním strojem.

Dále uvádím jen stručnou charakteristiku prostředí JAVA [9] , jsou zde uvedeny jen její specifikakterými se odlišuje od jiných programovacích jazyků na bázi C.

4. 1. 1. Struktura programu

Zdrojový text programu se skládá z jedné nebo více tříd a rozhraní, globální proměnné v Javě nejsou. Každá veřejná třída musí být uložena v samostatném zdrojovém souboru, jehož název je shodný s názvem této třídy a obsahuje příponu .java. Obdobné pravidlo platí i pro veřejná rozhraní. Na začátku každého souboru může být ještě uvedena deklarace balíku a/nebo import. Formát zdrojového souboru obecně vypadá takto:

```
package jménoBalíku ;
    import argument1;
    import argumentN;

    interface Rozhraní1 { /* tělo rozhraní 1 */ } ]
    interface RozhraníN { /* tělo rozhraní N */ } ]

   [ public class VeřejnáTřída { /* tělo veřejné třídy */ }
```

Všechny zdrojové soubory .java musí být uloženy v jednom adresáři, odkud bude proveden překlad.

4. 1. 2. Lexikální elementy

Java rozlišuje několik druhů lexikálních elementů: identifikátory, komentáře, konstanty, klíčová slova, operátory, oddělovače(Oddělovače jsou složené, kulaté a hranaté závorky, tečka, čárka a středník) a bílé znaky (Mezi bílé znaky patří mezera, tabulátor, přechod na další řádku atd.). Identifikátory, konstanty, klíčová slova a víceznakové operátory nesmí být rozděleny bílým

znakem. Na místě jednoho bílého znaku je možné uvést libovolný počet bílých znaků, což lze úspěšně použít pro přehledný zápis programu.

4. 1. 3. Základní datové typy

Základní datové typy jsou celočíselné, znakové, racionální a prázdný datový typ void, který se používá jen u metod.

Celočíselné typy:

název	bitů	rozsah
byte	8	-128..127
short	16	-32 768..32 767
int	32	-2 147 483 648..2 147 483 647
long	64	-9 223 372 036 854 775 808
		+9 223 372 036 854 775 807

Racionální typy:

název	bitů	rozsah
float	32	1.40129846432481707E-45
		3.40282346638528860E+38
double	64	4.94065645841246544E-324
		1.79769313486231570E+308

Logický typ:

Jediným logickým typem v Javě je boolean, pro který jsou definovány dvě hodnoty: false (logická 0) a true (logická 1). Výrazy, jejichž výsledkem je hodnota typu boolean, se používají v příkazech if, switch, while a for.

4. 1. 4. Balíky (packages)

Balíky (packages) slouží k vytváření nových prostorů jmen (namespaces) a umožňují tak například vytváření knihovnen tříd a rozhraní. Každý balík je obvykle reprezentován adresářem, který obsahuje přeložené třídy (soubory .class). Vnořováním adresářů vzniká hierarchie balíků.

Část hierarchie standardního Java Core API vypadá takto:

```
java
  |
  +--- applet
  +--- awt
  |     |
  |     +--- event
  |     +--- datatransfer
  +--- io
  +--- lang
  +--- net
  +--- ...
```

Jména adresářů odpovídají jménům balíků. Pro přístup ke třídám, statickým metodám a členským proměnným se v balících používá tečkové notace:

```
jménoBalíku.jménoTřídy
jménoBalíku.jménoTřídy.jménoMetody ( parametry )
jménoBalíku.jménoTřídy.jménoProměnné
```

Složená jména se oddělují tečkou (místo lomítek u adresářů). Jméno včetně balíku specifikuje úplné jméno.

Protože psaní úplných jmen v programu je zdlouhavé, umožňuje Java používat neúplná (jednoduchá) jména (simple names). Na začátku zdrojového souboru je potřeba rozšířit rozsah platnosti identifikátorů na požadované třídy a/nebo rozhraní. K tomu slouží klíčové slovo import:

```
import úplnéJménoTřídyNeboRozhraní ;
```

4. 1. 5. Výjimky

Výjimka (exception) je definována jako událost, která nastane během provádění programu a která naruší normální běh instrukcí. Výjimka je vyvolána například při chybném otevření souboru, při překročení mezí pole, při aritmetické chybě apod. Mechanismus výjimek umožňuje tyto chybové stavy zachytit a ošetřit. K tomu slouží dvojice bloků try a catch, jejichž syntaxe je následující:

```
try {
   // hlídaný blok
} catch( třídaVýjimek1 jménoProměnné1 ) {
   // ošetření výjimky
} catch( třídaVýjimek2 jménoProměnné2 ) {
   // ošetření výjimky
}
```

4. 1. 6. Vlákna (threads)

Java umožňuje tzv. multithreading neboli paralelní běh dvou či více částí programu. Této vlastnosti využije například Internetový prohlížeč - může nahrávat obrázky po síti, zároveň formátovat a zobrazovat WWW stránku a ještě k tomu spouštět applet - nebo textový editor, který může provádět kontrolu pravopisu, zatímco uživatel píše dokument apod. Každá paralelně běžící část programu se v Javě nazývá vlákno (thread). Důležité je, že jednotlivá vlákna lze naprogramovat téměř nezávisle a pouze v případě, že sdílí společná data nebo používají stejné prostředky (zařízení), je třeba zajistit jejich řádnou synchronizaci.

4. 2. Vývojové prostředí NetBeans

Pro programování v JAVĚ a pro tvorbu aplikace jsem si zvolil vývojové prostředí NetBeans IDE 6.0. Toto prostředí je volně šířené a dá se zdarma stáhnout z internetu[7]. Ještě před instalací NetBeans je potřeba nainstalovat balík JDK. Po úspěšné instalaci NetBeans a po jeho spuštění se můžeme pustit do tvorby projektu. V NetBeans IDE vyberte *File → New Project*. V okně *New Project wizard* vyberte category *JAVA* a tam *Java Application*. V poli *Project Name* zadejte Jméno aplikace. V poli *Create Main Class* zadejte jméno hlavní třídy. Nechejte políčko *Set as Main Project* zaškrtnuté. Klikněte na *Finish*. Projekt je vytvořen a otevřen v IDE. Pracovní plocha obsahuje několik oken:

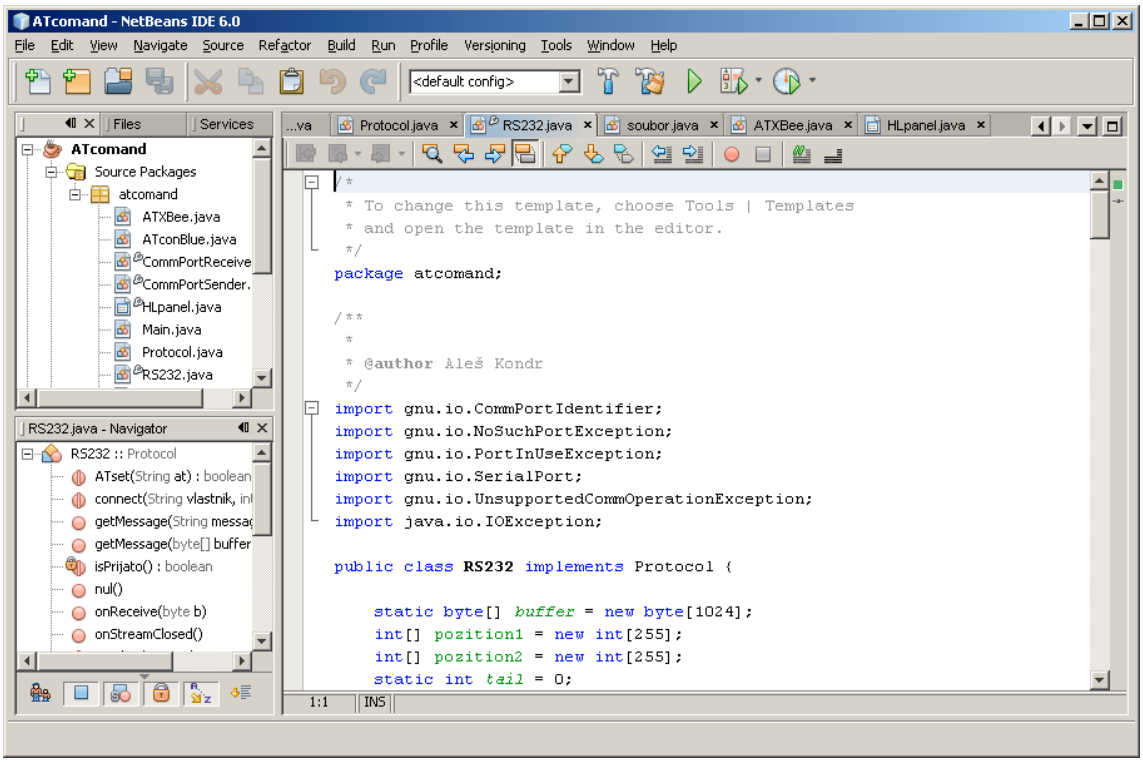
Okno Projects, umístěné vlevo nahoře, obsahuje stromový pohled na projekt zahrnující zdrojové kódy aplikace, knihovny, na kterých je projekt závislý a další. Toto okno může obsahovat více projektů.

Okno Navigator, které se nachází vlevo dole obsahuje elementy ve vybrané třídě, jako jsou metody a proměnné. Pomocí něj se můžeme mezi nimi rychle přepínat.

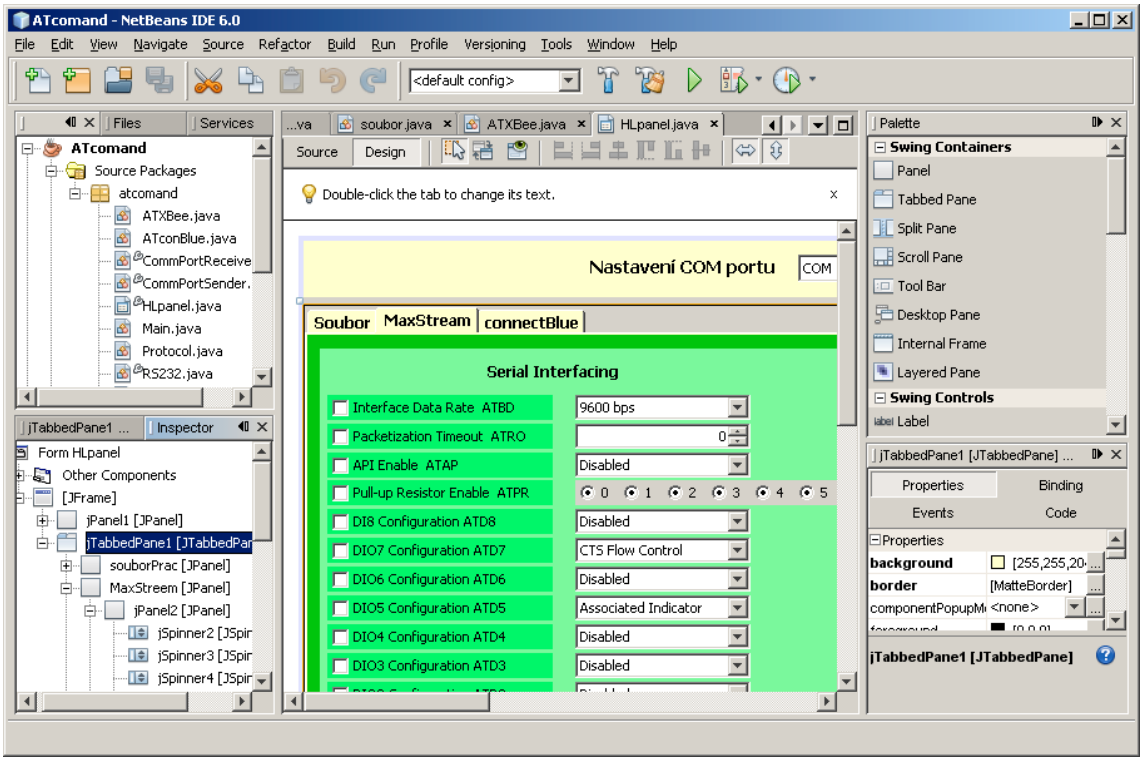
Okno Source Editor nacházející se vpravo přes celé okno obsahuje otevřený soubor příslušné třídy.

Pro přidání dalšího souboru klasické třídy do projektu vybereme *File → New File* . V okně *New File* vyberte *JAVA* category a tam *Java Class*. Vzhled takového okna nám zobrazuje Obr. 8.

Pro přidání dalšího souboru třídy grafického rozhraní GUI do projektu vybereme*: File* → *New File*. V okně *New File* vyberte *Swing GUI Forms* category a tam *jFrame Form*. Jak potom bude vypadat pracovní plocha NetBeans nám zobrazuje Obr. 9. Okno Source se nám změní z textového režimu na režim panelu, do kterého můžeme umisťovat jednotlivé grafické komponenty z okna Pallete, které se nově zobrazí vpravo nahoře. Vpravo dole se nově zobrazí okno *Properties*, s vlastnostmi vybrané grafické komponenty. Také okno *Navigátor* se změní na kaskádovité zobrazení jednotlivých komponent. Pro přepínání do textového režimu této třídy a zpět, slouží tlačítka *Source* a *Desing*, nacházející se vlevo nahoře okna *Source*.
Pokud chceme spustit námi vytvořený projekt, tak jej vyberem v okně *Project* a stiskem pravého tlačítka myši vybereme položku *Run*. Také můžeme použít zelené tlačítko šipky směrem doprava.



Obr. 8. Okno vývojového prostředí NetBeans IDE s otevřenou klasickou třídou.



Obr. 9. Prostředí NetBeans IDE s otevřenou třídou GUI pro tvorbu grafiky.

4. 3. Program pro žízení vložených zařízení pomocí AT příkazů

Mým cílem bylo vytvořit aplikaci, která se spustí na počítači s nainstalovaným operačním systému Windows a bude komunikovat se sériovým portem. K počítači se zároveň přes sériový port připojí modul, jehoš parametry chceme nastavit pomocí dávky AT příkazů do určité podoby. V aplikaci si jednoduchým systémem zaškrtání a nastavení grafických políček sestavíme požadovanou dávku AT příkazů a tu stiskem tlačítka odešleme do připojeného zařízení, čímž nastavíme jeho parametry na námi požadovanou hodnotu. Nyní vám popíši jednotlivé části mnou vytvořené JAVA aplikace.

4. 3. 1. Komunikace po COM

Pro komunikaci a posílání AT příkazů do připojeného zařízení se používá sériová komunikace RS232. Abychom mohli tuto komunikaci využít u naší aplikace, je potřeba do této apikace zaimplementovat patřičnou knihovnu. Takováto knihovna se dá najít a stáhnout zdarma z internetu [8], její název je *RXTXcomm.jar*. Je obsažena v balíku dalších potřebných souborů, z nich nejdůležitější jsou soubory pro podporu jednotlivých operačních systémů. Protože aplikaci programujeme pro operační systém Windows, tak nás bude zajímat především adresář se stejným názvem obsahující dva .dll soubory. Pro implementaci knihoven do operačního systému Windows musíme veškeré jmenované soubory zkopírovat do patřičných adresářů a to: Soubor *RXTXcomm.jar* vložíme do adresáře \jre\lib\ext . Soubory *rxtxSerial.dll* a *rxtxParallel.dll* do adresáře \jre\bin . Přičemž adresář \jre\ se většinou nachází na disku C:\ v C:\Program Files\Java\. Od této chvíle máme knihovnu RXTX přístupnou pro všechny aplikace vyvíjené v Javě.

V naší aplikaci slouží pro komunikaci přes com port tři třídy a to: *RS232*, *CommPortSender*, *CommPortReceiver*. První z nich s názvem *RS232* obsahuje asi nejdůležitější metodu *connect*:

```
public static void connect(String vlastnik, int port, int r)   {
try
{
String portName = "COM1";
int rychlost = 9600;
int databit = SerialPort.DATABITS_8;
int stopbit = SerialPort.STOPBITS_1;
int parity = SerialPort.PARITY_NONE;
portName = "COM" + String.valueOf(port);
CommPortIdentifier portIdentifier =
CommPortIdentifier.getPortIdentifier(portName);

switch (r) {
case 1:                 rychlost = 110;                 break;
case 2:                 rychlost = 300;                 break;
case 3:                 rychlost = 600;                 break;
case 4:                 rychlost = 1200;                 break;
case 5:                 rychlost = 2400;                 break;
case 6:                 rychlost = 4800;                 break;
case 7:                 rychlost = 9600;                 break;
case 8:                 rychlost = 14400;                break;
case 9:                 rychlost = 19200;                break;
case 10:                rychlost = 38400;                break;
case 11:                rychlost = 57600;                break;
case 12:                rychlost = 115200;               break;
case 13:                rychlost = 230400;               break;
case 14:                rychlost = 460800;               break;
case 15:                rychlost = 921600;               break;
            }
  if (portIdentifier.isCurrentlyOwned()) {
      new HLpanel().Chyba("COM port je používán!");
  } else {
      // nastavení vlastníka portu a timeout
     SerialPort serialPort = (SerialPort) portIdentifier.open(vlastnik, 2000);

      // nastavení parametrů pro připojení
      serialPort.setSerialPortParams(
              rychlost, databit, stopbit, parity);
      // nastavení zápisu na sériový port
      CommPortSender.setWriterStream(serialPort.getOutputStream());
      // nastavení čtení ze sériového portu
      new CommPortReceiver(serialPort.getInputStream()).start();
  }
}
```

Výpis kódu 4. Metoda pro připojení COM portu.

Tato metoda (Výpis kódu 4.) se při spuštění inicializuje třemi proměnnými. První proměnná tipu *String* obsahuje vlastníka obsazené sériové linky COM, což je vlastně jméno aplikace, která linku COM obsadí. Druhá proměnná tipu *integer* obsahuje číslo obsazené a používané linky COM, např. pokud do ní vložíme hodnotu 1 bude inicializována a používána linka COM1. Třetí proměnná také tipu *integer* slouží pro nastavení rychlosti komunikace námi zvolené sériové linky. Její hodnota se zadává v rozsahu 1 – 15 a odpovídají ji rychlosti 110 bit/s až 921600 bit/s. Při spuštění metody a inicializaci patřičné linky COM, je následně spuštěna

metoda *setWriteStream* v třídě *CommPortSender*, která v této třídě nastaví proměnnou *outputStream*. Také je ve třídě *CommPortReciever* spuštěno vlákno pro vyčítání příchozích dat přicházejících ze sériové linky COM.
Třída *CommPortSender* pro posílání dat na sériovou linku COM obsahuje dvě metody s názvem *send* a *send2*:

```
public static void send2(String bytes2){
CommPortSender.send(new RS232().getMessage(bytes2));
}
public static void send(byte[] bytes) {
    try {
      // poslaní na com port je jednoduše odesláno do OutputStream
        out.write(bytes);
        out.flush();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Výpis kódu 5. Metoda pro odesílání pole znaků na COM port.

Metoda *send2* (Výpis kódu 5.) se inicializuje proměnnou pole tipu *bytes,* které je odesláno na COM port a začne se vysílat směrem ven. Pokud však potřebujeme odeslat proměnnou tipu *String*, musíme jí inicializovat metodu *send2,* která řetězec převede na pole bajtů a odešle do metody *send*.

Třída *CommPortReciever* slouží pro vyčítání dat přicházejících na COM port. Obsahuje metodu *run,* což je vlastně vlákno spouštěné při inicializaci COM portu a běžící nezávisle na chodu programu:

```
public void run() {
        try {
            int b;
            while(true) {
     // pokud z metody in.read() nepřichází datový tok metoda vrací hodnotu -1
                while((b = in.read()) != -1) {
                    protocol.onReceive((byte) b);
                }
                protocol.onStreamClosed();
                // počká 10ms a zkusí znovu
                sleep(10);
            }
```

Výpis kódu 6. Metoda spouštěná jako vlákno nezávisle na chodu programu.

Toto vlákno (Výpis kódu 6.) je vlastně uzavřená smyčka, která opakovaně čte znaky přicházející z portu. Pokud z portu nepřichází žádný znak, tak volání metody *read()* vrací hodnotu –1. V tomto případě je volána metoda *onStreamClosed,* následně se vyčká 10ms a stav portu je opětovně kontrolován. Pokud přijde na vstupu nějaký znak nebo skupina znaků, tyto jsou vyčítány metodou *read()* a po převedení do proměnné *byte* je jimi inicializována metoda *onReceive*. Metody *onReceive* a *onStreamClosed* se nachází v třídě *RS232* a jsou pomocí Interface s názvem *protokol* spojeny s třídou *CommPortReciever*.

```
public void onReceive(byte b) {
```

```
  stav = true;
    buffer[tail] = b;
    tail++;
    if ((b == znak) && zjist) {
        tpoz2++;
        pozition2[tpoz2] = tail - 1;
    }
}

public void onStreamClosed() {
    if (stav) {
        stav = false;
        tpoz1++;
        pozition1[tpoz1] = tail - 1;
        setPrijato (true);
        if (tail > 1000) {
            tail = 0;
        }
    }
}
```

Výpis kódu 7. Metody na zachycení příjmu z COM portu a jeho ukončení.

Metoda *onReceive* (Výpis kódu 7.) přijatý znak uloží do pole tipu *byte* s názvem *buffer*. Následně ukazatel *tail,* který ukazuje na poslední přijatý znak v poli zvýší o jedničku. To způsobí, že se postupně ukládají přijímané znaky do pole *buffer*. Pokud je boolean proměnná s názvem *zjist* ve stavu *true* a pokud je přijatý znak shodný jako znak námi přednastavený uložený v proměnné *znak*, tak se provede uložení jeho pozice do pole s názvem *pozition2*. Toho se dá využít v případě, že potřebujeme v poli přijatých znaků znát pozice určitého znaku.

Druhá metoda s názvem *onStreamClosed* reaguje na ukončení příjmu skupiny znaků. Toto si můžeme představit jako mezery mezi slovy v případě řeči. Přijímané znaky jsou jednotlivá písmena která vyslovujeme a ze kterých se skládá slovo. V případě vynechání písmene, v našem případě znaku, nastává právě taková mezera, na kterou reaguje výše zmiňovaná metoda. Aby se metoda neprováděla opakovaně, je do ní umístěn stavový automat s boolean proměnnou *stav*, který způsobí, že se metoda provede jen jednou v období mezery. V těle této metody je umístěno stejně jako v předchozí metodě ukládání pozice pole buffer do pole *pozition1*. Také je zde umístěno volání a nastavení synchronizační metody *setPrijato(true).*

```
synchronized static private boolean isPrijato ()
     {
     return prijato;
     }

synchronized static private void setPrijato (boolean val)
     {
     prijato = val;
     }
```

Výpis kódu 8. Metody pro synchronizaci vlákna s programem.

Synchronizační metody *isPrijato* a *setPrijato* (Výpis kódu 8.) slouží pro nastavování a čtení proměnné prijato. Pro odesílání samotných AT příkazů slouží metoda *Atset*.

```
public static boolean ATset(String at) {
```

```
try
{
boolean atc = true;
setPrijato (false);
tail = 0;
CommPortSender.send2("AT" + at + "\r\n");
if (isPrijato ()) {
    String radek = new String(buffer, 0, tail);
    if ((radek.indexOf("OK") != -1) || (radek.indexOf("ok") != -1)) {
        atc = false;
    }
}
if (atc) {
    Thread.sleep(10);
    if (isPrijato ()) {
        String radek = new String(buffer, 0, tail);
       if ((radek.indexOf("OK") != -1) || (radek.indexOf("ok") != -1)) {
            atc = false;
        }
    }
}
if (atc) {
    Thread.sleep(20);
    if (isPrijato ()) {
        String radek = new String(buffer, 0, tail);
       if ((radek.indexOf("OK") != -1) || (radek.indexOf("ok") != -1)) {
            atc = false;
        }
    }
}
if (atc) {
    Thread.sleep(40);
    if (isPrijato ()) {
        String radek = new String(buffer, 0, tail);
       if ((radek.indexOf("OK") != -1) || (radek.indexOf("ok") != -1)) {
            atc = false;
        }
    }
}
return atc;
}
```

Výpis kódu 9. Metoda odeslání AT příkazu.

Metoda *ATset* (Výpis kódu 9.) je inicializována proměnnou tipu *String,* obsahující AT příkaz bez počátečních písmen AT, ty jsou dodány v metodě, také jsou dodány znaky <CR> a <LF>. Takže například odeslání příkazu "ATMYac50<CR><LF>" se provede tak, že se metoda inicializuje řetězcem "MYac50". Zpět metoda vrací proměnnou tipu boolean, která nám značí, jestli byl příkaz přijat koncovým zařízením. To poznáme tak, že se zpět po COM portu vrátí od zařízení sled znaků "OK". Pokud je vrácená hodnota ve tvaru false, tak zařízení vrátilo znaky "OK" a odeslání AT příkazu je v pořádku. Pokud je ve tvaru true, tak zařízení buťto nevrátilo žádné znaky, nebo znaky které neobsahují sekvenci "OK".

Poslední důležitá metoda která je obsažená v třídě *RS232* je *sendstr*. Ta funguje podobně jako *Atset*. Také se inicializuje proměnnou string která je odeslána na COM port, avšak

v nezměněném tvaru, nic se do ní nepřidává. Zpět vrací také proměnnou String s obsahem toho, co na COM port přišlo zpět od cílového zařízení.

4. 3. 2. Práce se soubory

Pro práci se soubory slouží v mém programu třída s názvem *soubor*. Soubory zde využívám jako dočasné úložiště dávky AT příkazů, které se mají odeslat na COM port. A dalších podpůrných sekvencí, jako například escape sequence, sloužící pro nastavení modulu tak, aby příjmal AT příkazy. Základní soubory které zde využívám je: *atcom.atc,* soubor což je soubor kam se uloží dávka At příkazů těsně před odesláním na COM port a pak soubor *error.atc,* kam se odkládají veškeré AT příkazy, které se nepodařilo doručit cílovému zařízení.
Pro odesílání AT příkazů ze souboru na COM port slouží metoda nazvaná *sendAT*.

```
public static void sendAT() {
    try {
        RandomAccessFile zdroj = new RandomAccessFile("atcom.atc", "r");
        new File("error.atc").delete();
        RandomAccessFile chyby = new RandomAccessFile("error.atc", "rw");
        String radek;
        zdroj.seek(0);
        radek = zdroj.readLine();
        if (radek.indexOf("ESEQ:") != -1) {
  String message = RS232.sendstr(radek.substring(radek.indexOf("ESEQ:") + 5));
         if ((message.indexOf("OK") != -1) || (message.indexOf("ok") != -1)) {
                while ((zdroj.getFilePointer() + 2) <= zdroj.length()) {
                    radek = zdroj.readLine();
                    if (radek.indexOf("AT") != -1) {
                  if (RS232.ATset(radek.substring(radek.indexOf("AT") + 2))) {
                            chyby.writeBytes(radek + "\r\n");
                        }
                    }
                }
            } else {
               new HLpanel().Chyba("Nepodařil se vstup do modu AT příkazů!");
            }
        } else {
            new HLpanel().Chyba("Soubor neobsahuje ESCAPE sekvenci!");
        }
    } catch (IOException ex) {
        new HLpanel().Chyba("Nastala chyba pri čtení souboru!");
    }
}
```

Výpis kódu 10. Metoda odeslání dávky ST příkazů ze souboru.

Metoda (Výpis kódu 10.) nastaví jako zdroj AT příkazů soubor *atcom.atc*, smaže soubor *error.atc* který mohl obsahovat z minula nějaké chybně odeslané AT příkazy a vytvoří nový soubor a přiřadí jej jako chyby. V souboru nastaveném jako provede skok na pozici 0 a do String proměnné *radek* uloží první řádek ze zdroje, tj. od aktuální pozice až do pozice se znaky <CR><LF>. Protože se nacházíme na prvním řádku zdroje, kde by se za sekvencí znaků "ESEQ:" měla nacházet escape sequence (sekvence znaků pro vstup připojeného modulu do módu AT příkazů), provede se její kontrola. Pokud se zde escape sequence opravdu nachází, odešlou se znaky které obsahuje, pomocí metody *sendstr* na COM port. Vrácená proměnná String z metody *sendstr* se zkontroluje zdali obsahuje sekvenci "OK", tj. jestli se připojený modul

opravdu nachází v módu pro zadávání AT příkazů. Pokud ano, může nastat vyčítání dávky AT příkazů a jejich odesílání na COM port. To se děje smyčkou *while,* která probíhá tak dlouho, dokud ukazatel na pozici ve zdroji zvýšený o hodnotu 2 není větší nebo roven celkové délce zdroje. Ve smyčce se přesune řádek ze *zdroje* do proměnné *radek*, ten se následně zkontroluje na obsah sekvence AT. Pokud ji obsahuje, tak veškeré následující znaky až do konce řádku se odešlou metodou *ATset* na COM port. Pokud se zadání AT příkazu nezdaří a metoda vrátí hodnotu true, je tento řádek zapsán do souboru *chyby*.
Pro přesun nějakého cizího souboru obsahujícího AT příkazy do souboru *atcom.atc* slouží metoda se jménem *otevriAT*.

```
public static void otevriAT(String soubor) {
    try {
        RandomAccessFile zdroj = new RandomAccessFile(soubor, "r");
        RandomAccessFile cil = new RandomAccessFile("atcom.atc", "rw");
        String radek;
        while ((zdroj.getFilePointer() + 2) <= zdroj.length()) {
            radek = zdroj.readLine();
            if (radek.indexOf("AT") != -1) {
   cil.writeBytes("AT" + (radek.substring(radek.indexOf("AT") + 2)) + "\r\n");
            }
            if (radek.indexOf("ESEQ:") != -1) {
cil.writeBytes("ESEQ:" + (radek.substring(radek.indexOf("ESEQ:") + 5)) +
"\r\n");
            }
        }
    } catch (IOException ex) {
        new HLpanel().Chyba("Nastala chyba pri kopírování souboru!");
    }
}
```

Výpis kódu 11. Metoda pro kopiji AT příkazů z jednoho souboru do druhého.

Tato metoda (Výpis kódu 11.) se inicializuje proměnnou String, ve které je obsažena cesta ke kopírovanému souboru, ten se následně označí jako *zdroj* a náš soubor *atcom.atc* se označí jako *cíl*. Jediné co se zkopíruje jsou řádky obsahující sekvenci “AT“ a sekvenci “ESEQ:“ a to tím způsobem, že se zkopíruje na těchto řádcích vše, od těchto sekvencí až po konec řádku, tj. až po znaky <CR><LF>.

Skoro stejně jako *otevriAT*, funguje také metoda *presunAT,* akorát s tím rozdílem, že je zaměněn *zdroj* a *cíl,* tj. přenos jde ze souboru *atcom.atc* do souboru určeného inicializační proměnnou. Podobně jako předchozí dvě metody funguje ještě metoda *ulozAT*, ta je však inicializována dvěma proměnnými string, přičemž přesun se provádí ze souboru určeného první z nich, do souboru který určuje ta druhá.
Pro vytváření nějakého souboru AT příkazů slouží metody *zapisAT* a *zapisESEQ*.

```
public static void zapisAT(int poz, String ret, String soubor) {
        try {
            RandomAccessFile f = new RandomAccessFile(soubor, "rw");
            f.seek((poz * 20)+20);
            f.writeBytes("AT" + ret + "\r\n");
        } catch (IOException ex) {
            new HLpanel().Chyba("Nastala chyba pri zápisu souboru!");
        }
    }
```

Výpis kódu 12. Metoda pro zápis AT příkazu do souboru.

Metoda *zapisAT* (Výpis kódu 12.) se inicializuje třemi proměnnými. První je tipu integer a udává pozici v souboru v násobcích hodnoty 20, kam se má uložit sekvence znaků, které udává druhá inicializační proměnná tipu string. Třetí inicializační proměnná tipu string nám udává umístění souboru, do kterého se má ukládat.

```
public static void zapisESEQ(String soubor, String ret) {
        try {
            RandomAccessFile f = new RandomAccessFile(soubor, "rw");
            f.seek(0);
            f.writeBytes("ESEQ:" + ret + "\r\n");
        } catch (IOException ex) {
            new HLpanel().Chyba("Nastala chyba pri zápisu souboru!");
        }
    }
```

Výpis kódu 13. Metoda pro zápis escape seqence do souboru.

Metoda *zapisESEQ* (Výpis kódu 13.) slouží pro zápis escape sequence na nulovou pozici do souboru. Inicializuje se dvěma proměnnými string. První nám udává umístění souboru a druhá obsahuje znaky escape sequence.

4. 3. 3. Převádění čísel

Pro takový převod čísel, pro který nenajdeme v JAVĚ žádnou již hotovou metodu, slouží třída *prevod*. V té se nachází netradiční metody pro převod čísel mezi různými tipy proměnných a mezi různými formáty. Zatím jediná taková metoda která byla v naší aplikaci potřeba, je metoda *hexa*. Slouží pro převod čísla tipu ineger, do proměnné tipu string, která obsahuje řetězec znaků hexadecimálního čísla vstupní proměnné, tj. pouze znaky 0-9,a,b,c,d,e,f .

```
public class prepocet {
    public String hexa(int cislo) {
        int a;
        String c = "";
        String d = "";
        do {
            a = cislo % 16;
            cislo = cislo / 16;
            if (a >= 0 && a <= 9) {
                c = String.valueOf(a);
            } else {
                switch (a) {
                    case 10:
                        c = "a";
                        break;
                    case 11:
                        c = "b";
                        break;
                    case 12:
                        c = "c";
                        break;
                    case 13:
                        c = "d";
                        break;
```

```
                    case 14:
                        c = "e";
                        break;
                    case 15:
                        c = "f";
                        break;
                }
            }
            d = c + d;
        } while (cislo > 0);
        return d;
    }
}
```

Výpis kódu 14. Metoda pro převod čísel.

Metoda (Výpis kódu 14.) se inicializuje proměnnou integer obsahující nějaké kladné číslo. Toto číslo se dělí hodnotou 16 tak dlouho, dokud jeho podíl není roven 0. Zároveň se také určuje zbytek po dělení 16 a podle něj se určuje znak na patřičném místě. Pokud je zbytek dělení v rozsahu 0 – 9, tak se toto číslo pouze převede na znak proměnné string. Pokud je číslo v rozsahu 10 – 15, tak se jako znak string zadá písmeno a – f. Nakonci se znaky poskládají za sebe a metoda takovýto řetězec vrátí zpět.

4. 3. 4. Sestavení AT příkazu pro konkrétní modul

Pro jednotlivé druhy modulů (v mém případě moduly MaxStream a connectBlue), jsou v mé aplikaci obsaženy třídy *ATXBee* a *ATconBlue*, které obsahují metody pro sestavení konkrétních AT příkazů. Vstupem takovýchto metod jsou proměnné, které má daný AT příkaz obsahovat. Vstupem jsou proměnné tipu string, obsahující kompletní AT příkaz i z doprovodným parametrem. Pro možnost přímého uložení každého příkazu do souboru, jsem třídu opatřil dvěma statickými proměnnými, sloužící jako konstanty. Jsou to string proměnná s názvem *kam* obsahující jméno souboru, do kterého se má ukládat výsledná dávka AT příkazů a proměnná tipu boolean s názvem *uloz*, která nám ukládání příkazu povoluje.
Pro příklad uvedu třeba metodu zpracovávající příkaz ATCH který nastavuje na modulech MaxStream číslo komunikačního kanálu:

```
public String ATCH(int a) {
        String prikaz = "CH" + new prepocet().hexa(a + 11);
        if (uloz) {
            soubor.zapisAT(1, prikaz, kam);
        }
        return prikaz;
    }
```

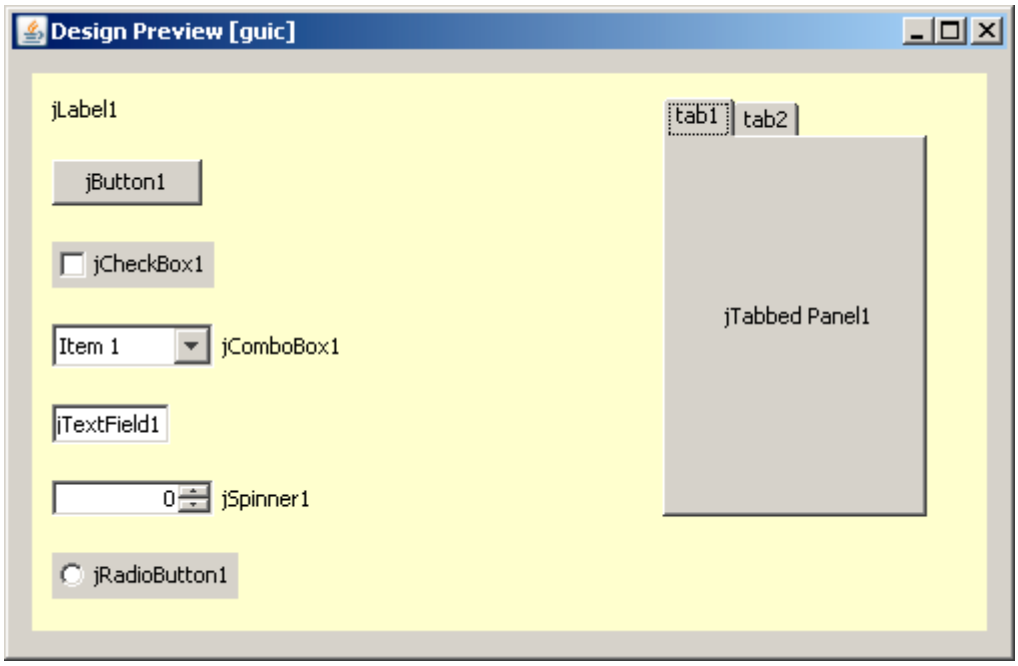
Výpis kódu 15. Metoda vytvářející AT příkaz.

Metoda (Výpis kódu 15.) je inicializována proměnnou integer, udávající číslo kanálu v rozsahu 1 – 15. Tato hodnota se zvíší o hodnotu 11, převede se metodou hexa na hexadecimální řetězec string a připojí se k ní znaky "CH". Pokud je statickou proměnnou *uloz* povoleno ukládání rovnou do souboru, tak se celý řetězec uloží na pozici 1, do souboru s názvem daným proměnnou *kam*. Zároveň se také celý řetězec nastaví jako vracená proměnná metody.
Kromě metod sestavujících AT příkazy obsahuje každá třída metody *smaz*, *presun*, *uloz*.
- Metoda *smaz* provádí vymazání souboru s názvem daným proměnnou *kam* a následným

vytvořením nového souboru se stejným názvem, do nějž se zapíše escape sekvenci metodou *zapisESEQ*.
- Metoda *presun* provádí zkopírování souboru daném proměnnou *kam*, do souboru *atcom.atc*, určeného jako mezi skladiště dávky, určené pro odeslání na COM port.
- Metoda *uloz* se inicializuje proměnnou obsahující umístění ukládaného souboru AT příkazů. Do tohoto souboru se zkopíruje obsah souboru s názvem daném proměnnou *kam*.

4. 3. 5. Vzhled a grafické rozhraní mojí aplikace (GUI)

Veškeré prvky grafického rozhraní mojí aplikace se nachází ve třídě s názvem *HLpanel*. Při vytváření této třídy ji vytvoříme jako specielní třídu tipu *Swing GUI Forms – JFrame Form*. To nám umožní, že nebude vytvořena jako klasická třída pro zápis zdrojového kódu, ale jako okno, do kterého můžeme vkládat jednotlivé prvky grafického rozhraní. Mnou použité prvky jsou Obr. 10:



Obr. 10. Mnou použité ovládací prvky grafického rozhraní.

jPanel - Panel představuje nejjednodušší typ kontejnerové komponenty. Jeho jediným úkolem je organizovat komponenty, které jsou v něm obsaženy. Panely se používají při návrhu vzhledu okna. Za použití vhodného layout manageru umístíme na formulář panely, na tyto panely následně další komponenty.

jLabel - Tato komponenta je používána jako popis. Komponentě lze nastavit barvu popředí nebo barvu pozadí. Většinou je používána jako komponenta statická, kdy zpravidla neošetřujeme jeho události, ale pouze s ním popisujeme jiné komponenty. Základní nejpoužívanějších metody spolu se stručným popisem jsou:
setText() - Nastavení textu zobrazovaného labelem
getText() - Získání textu zobrazovaného labelem.

jButton - neboli tlačítko představuje jednu z nejpoužívanějších komponent, zpravidla slouží ke spouštění nějaké akce. Tlačítko po svém stisknutí generuje událost, kterou je nutné odchytit prostřednictvím rozhraní ActionListener za použití metody actionPerformed.

JCheckBox - představuje zaškrtávací pole, může nabývat logické hodnoty true/false (realizované zaškrtnutím/odškrtnutím). CheckBoxy je možno seskupovat, komponenty jsou na sobě nezávislé; mohou být zaškrtnuty/odškrtnuty v libovolné kombinaci. Při změně stavu checkboxu je generována událost, kterou lze odchytit prostřednictví rozhraní ItemListener za použití metody itemStateChanged(). Základní nejpoužívanějších metody spolu se stručným popisem jsou:
setSelected(boolean status) - Nastavení stavu checkboxu.
isSelected() - Získání stavu checkboxu.

JComboBox - představuje rozbalovací seznam. Seznam je tvořen jednotlivými položkami setříděných podle předem známého klíče. Viditelná je pouze jedna řádka, po kliknutí na šipku se seznam rozbalí a umožní zvolit kliknutím některou z položek seznamu. Použití je podobné jako v případě radio buttonů, komponenta je vhodná pro výběr z většího množství variant. Metoda actionPerformed() je vyvolána v okamžiku, kdy uživatel zvolí některou z položek combo boxu nebo po editaci některé položky stiskne klávesu Enter Základní nejpoužívanějších metody spolu se stručným popisem jsou:
getSelectedIndex() – Získá index označené položky.
setSelectedIndex() - Nastavení požadované položky seznamu.
getSelectedItem() - Získání textu označené položky.

jTextField - Komponenta představuje editační políčko, do kterého mohou být zadávány jednořádkové údaje z klávesnice. Potvrzení údajů je provedeno stiskem klávesy Enter. Zadávaný text může být delší než šířka JTextFieldu. Komponenta se používá, pokud potřebujeme od uživatele získat vstupní údaje nutné pro běh programu. Po stisku klávesy Enter je generována událost, kterou lze odchytit prostřednictví rozhraní ActionListener za použití metody actionPerformed().Základní nejpoužívanějších metody spolu se stručným popisem jsou:
getText() – Získá text který je obsažen v políčku
setText() – Nastaví text do políčka

jSpinner – Tato komponenta slouží pro vkládání nějaké číselné hodnoty. Na boku komponenty jsou dvě tlačítka s označením šipek nahoru a dolu pomocí nichš se dá hodnota inkrementovat nebo dekrementovat. Při změně hodnoty je generována událost, kterou lze odchytit prostřednictví rozhraní ChangeEvent za použití metody StateChanged().Základní nejpoužívanějších metody spolu se stručným popisem jsou:
getValue() – Získá číselnou hodnotu spinneru
setValue() - Nastaví číselnou hodnotu do spinneru

jTabbledPanel – Tato komponenta slouží pro přepínání mezi více panely, které jsou umístěny na sobě. Přepínač má formuzáložek, na které když se cvakne, tak se zobrazí příslušný panel.

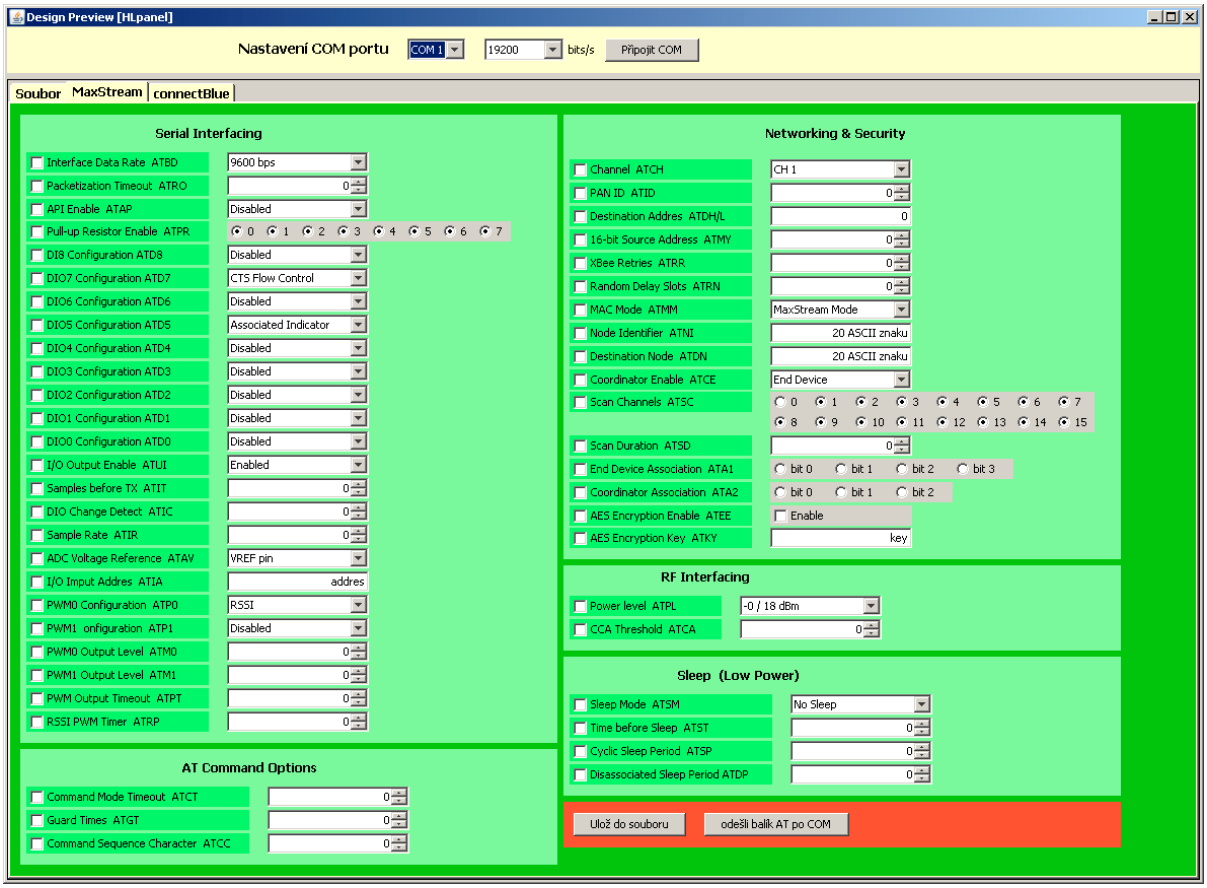
4. 3. 6. Jak moje aplikace pracuje

Po spuštění aplikace se spustí metoda *main* umístěná ve třídě *Main.*

```
public static void main(String[] args) {
                new HLpanel().setVisible(true);
                  }
```

Výpis kódu 16. Metoda spouštěná po spuštění aplikace.

Jediné co tato metoda (Výpis kódu 16.) provede je, že se spustí a zobrazí *Hlpanel*, což je grafické okno aplikace. Veškeré ostatní funkce si už uživatel spouští sám stisky jednotlivých prvků v tomto okně.



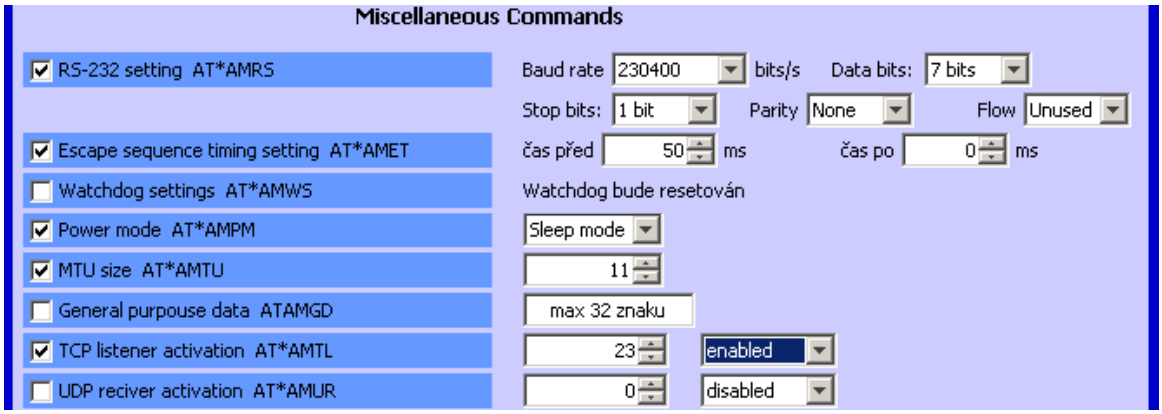
Obr. 11. Vzhled aplikace.

Toto okno viz. Obr. 11 , obsahuje několik panelů s umístěnými ovládacími prvky. Na úzkém panelu na horní straně aplikace se nachází několik prvků, pomocí nichš provedeme připojení k COM portu. Jsou to dva ComboBox prvky, ve kterých se nastavuje který port má být využívaný a jakou má pracovat rychlostí, dále je zde Button, při jehoš stisku se provede metoda (Výpis kódu 17.):

```
private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
RS232.connect("AT comand", (jComboBox4.getSelectedIndex())+1 ,
(jComboBox1.getSelectedIndex())+1
    }
```

Výpis kódu 17. Metoda reagující na stisk tlačítka pro připojení k COM portu.

Která vyčte z ComboBoxů číslo portu a rychlost a těmito hodnotami inicializuje metodu *connect*, sloužící pro inicializaci COM portu. Jméno vlastníka portu je nastaveno na "AT comand".Pod tímto panelem se nachází TabbledPanel, který v sobě združuje více přes sebe překrytých panelů. Zobrazení jednotlivých panelů se provádí pomocí záložek umístěných vlevo nahoře. Na každém z těchto panelů je umístěna skupina AT příkazů pro jednotlivý modul.



Obr. 12. Vzhled grafiky jednotlivých AT příkazů

Každý AT příkaz je reprezentován jedním prvkem CheckBox a dalšími nastavovacími prvky Obr. 12. Všechny nastavovací prvky mají vazbu na CheckBox realizovánu metodou *ActionPerformed*:

```
private void jComboBox31ActionPerformed(java.awt.event.ActionEvent evt) {
     jCheckBox36.setSelected(true);
 }
```

Výpis kódu 18. Metoda pro nastavení CheckBoxu na hodnotu true.

Takže pokud se změní vlivem uživatele jejich hodnota, tak se CheckBox nastaví na hodnotu true(Výpis kódu 18.).
Vyčítání hodnot nastavovacích prvků a jejich ukládání do souboru je realizováno private metodou která je umístěna v třídě Hlpanel.

```
private void ulozXB (){
        new ATXBee().smaz();
if (jCheckXB3.isSelected()) new ATXBee().ATDHL(jTextField3.getText());
if (jCheckXB11.isSelected()) new ATXBee().ATNI(jTextField5.getText());
if (jCheckXB14.isSelected()) new ATXBee().ATDN(jTextField6.getText());
if (jCheckXB27.isSelected()) new ATXBee().ATPL(jComboBox2.getSelectedIndex());
if (jCheckXB29.isSelected()) new ATXBee().ATSM(jComboBox3.getSelectedIndex());
```

Výpis kódu 19.

Tato metoda (Výpis kódu 19.) po zpuštění smaže soubor AT příkazů, který odpovídá příslušnému modulu. Potom následuje soustava příkazů *if*, které zjišťují, zdali jsou zaškrtnuty jednotlivé CheckBoxy příslušející jednotlivým příkazům. Pokud ano, jsou vyčteny hodnoty nastavovacích prvků příkazů a je jimi inicializována metoda pro patřičný AT příkaz. Takže dojde k vytvoření dávkového souboru pro konkrétní modul, který obsahuje jen ty příkazy, které jsou zaškrtnuty.

Dále jsou na každém panelu AT příkazů pro jednotlivý modul umístěny dvě tlačítka tipu Button s popisky: *Ulož do souboru* a *Odešli balík AT po COM*. Při stisku prvního se spustí metoda (Výpis kódu 20.):

```
private void jButton9ActionPerformed(java.awt.event.ActionEvent evt) {
    ulozXB();
    JFileChooser fileChooser = new JFileChooser();
    int retval = fileChooser.showSaveDialog(null);
    if (retval == JFileChooser.APPROVE_OPTION) {
        File file = fileChooser.getSelectedFile();
         new ATXBee().uloz("" + file);
        }
    }
```

Výpis kódu 20. Metoda pro uložení do souboru

Která vytvoří dávkový soubor zaškrtnutých AT příkazů. Poté je zobrazeno okno FileChooser, ve kterém si můžeme vybrat adresář a soubor do kterého se má uložit dávka AT příkazů. Po stisku tlačítka Save, se string řetězcem obsahujícím cestu k souboru inicializuje metoda *ulož*. Ta způsobí uložení dávky AT příkazů.
Další Button s názvem Odešli balík AT po COM (Výpis kódu 21.) při stisku způsobí.

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        ulozXB();
        new ATXBee().presun();
        soubor.sendAT();
    }
```

Výpis kódu 21. Metoda odeslání na COM port

Vytvoření dávkového souboru zaškrtnutých AT příkazů. Přesunutí do souboru pro odeslání AT příkazů a jejich odeslání na COM port.

ZÁVĚR

Moje práce se zabývá problematikou AT příkazů a jejich využití, při komunikaci s vloženým bezdrátovým zařízením. Pro bezdrátovou komunikaci jsou zde představeny dva moduly. Konkrétně jsou to moduly pro komunikaci standardem ZigBee a WiFi, ke kterým jsou zde uvedeny základní parametry. Protože veškerý servis, nastavování a zjišťování parametrů a stavů takovýchto modulů je prováděn prostřednictvím AT příkazů, je zde uveden jejich kompletní seznam, se stručným popisem každého z nich.

Pro usnadnění práce s AT příkazy a pro automatizaci procesu měření a testování komunikačních zařízení jsem ~~si~~ zvolil nástroj Tera Term. Jde o program pro operační systém Windows, podobný nástroji Hyperterminal. Oproti němu však disponuje vlastností zadávání tzv. macro souborů, které dokáží komunikaci prostřednictvím AT příkazů zautomatizovat. Protože však tento nástroj (program) není mezi veřejností příliš znám a využíván, vytvořil jsem pro něj stručnou příručku. Tu ocení především ti, kteří budou potřebovat jednoduchý automatický nástroj schopný komunikovat nejen po sériové lince. Také bylo vytvořeno několik ukázkových macro souborů, které provádějí měření a testování výše uvedených modulů. Nástroj Tera Term má však pro danou aplikaci několik nedostatků, proto jsem jej pro můj další úkol dále nepoužil. Uvádím jej zde však proto, že jde o nástroj jednoduchý, programátorsky rychle zvládnutelný a pro jednoduché aplikace plně dostatečný.

Nedostatky Tera Termu jsou především absence komplexnějšího grafického rozhraní, komunikujícího s uživatelem. Tuto nectnost plně nahradí další nástroj, programovací jazyk JAVA. V tomto nástroji byla vytvořena aplikace pro rychlé a snadné řízení vložených zařízení pomocí AT příkazů. Tato aplikace zajišťuje jednoduchý a graficky přehledný výběr souboru AT příkazů, potřebných pro nastavení výše popsaných modulů do námi požadovaného stavu. Dále zajišťuje sestavení dávky AT příkazů a její odeslání po sériové lince do modulu. Pro potřeby opakování nastavovacího procesu, je zde možnost soubor AT příkazů uložit na disk a kdykoliv použít znovu. V případě potřeby je možné do mé aplikace snadno implementovat další modul, komunikující pomocí AT příkazů. Tato aplikace bude nejen velkým přínosem pro firmu Betacontrol, pro niž jsem ji vytvářel, ale i po všechny, kdož vyžadují jednoduše a rychle nastavovat popisovaná zařízení.

Seznam použité literatury

[1] PATRICK, D. Wireless Network Coexistence. McGraw-Hill Professional Publishing, 2004. ISBN 0071399151

[2] MC13192/MC13193 2.4 GHz Low Power Transciever for IEEE 802.15.4 Standard. Freescale Semiconductor, 2005. MC13192

[3] XBee/XBee-PRO OEM RF Modules Product Manual – IEEE 802.15.4 OEM RF Modules MaxStream, 2006, M100232

[4] Webové stránky Fy. MaxStream: URL: <http://www.maxstream.net> [cit. 2006-28-12].

[5] CONNECTBLUE OWSPA311G I/X CB-0908, AT Comannds set, CONNECTBLUE AB, 2007, cBProject-0604-02 (1)

[6] Webové stránky Programu. Tera Term: URL: <http://ttssh2.sourceforge.jp/> [cit. 2007-7-5].

[7] Webové stránky Programu. NetBeansIDE 6.0: URL: < http://www.netbeans.org > [cit. 2008-26-5].

[8] Webové stránky knihovny. RXTX: URL: < http://users.frii.com/jarvi/rxtx/index.html > [cit. 2008-26-5].

[9] KOTALA, Zdeněk. TOMAN, Petr. Programovací jazyk Java. Západočeská univerzita Fakulta aplikovaných věd Katedra informatiky a výpočetní techniky. Sborník [online]. 2001 [cit. 2008-26-5]. Dostupné z WWW: < http://dione.zcu.cz/java/sbornik.html >.

Seznamy zkratek a symbolů

DSSS - *Direct Sequence Spread Spectrum* technika přímého rozprostřeného spektra. Je jednou z metod pro rozšíření spektra při bezdrátovém přenosu dat

QPSK - *Quadrature Phase Shift Keying* je čtyřstavová digitální modulace, u které se mění pouze fáze.

API - *Application programming interface* což znamená rozhraní pro programování aplikací.

PWM - *Pulse Width Modulation* je proces, při kterém dochází ke změně šířky pulsu nějakého nosného signálu

WiFi - *Wireless Fidelity* je standard pro lokální bezdrátové sítě (Wireless LAN, WLAN) a vychází ze specifikace IEEE 802.11.

ADC - *Analog digitál convertor* je elektronická součástka určená pro převod spojitého (neboli analogového) signálu na signál diskrétní (neboli digitální).

SSID - *Service Set Identifier* je jedinečný identifikátor každé bezdrátové (WiFi) počítačové sítě.

GUI - *Graphical User Interface* je druh komunikace s počítačem mající podobu interaktivních grafických prvků.

JDK - *Java Development Kit* je soubor základních nástrojů pro vývoj aplikací pro platformu Java.

DHCP - *Dynamic Host Configuration Protocol* je aplikační protokol z rodiny TCP/IP. Používá se pro automatické přidělování IP adres koncovým stanicím v síti.

DNS - *Domain Name Systém* je hierarchický systém doménových jmen, který je realizován servery DNS a protokolem stejného jména, kterým si vyměňují informace. Jeho hlavním úkolem a příčinou vzniku jsou vzájemné převody doménových jmen a IP adres uzlů sítě.

MAC adresa - *Media Access Control* je jedinečný identifikátor síťového zařízení, který používají různé protokoly druhé (spojové) vrstvy OSI. Je přiřazována síťové kartě NIC bezprostředně při její výrobě.

COM port – Označení sériové linky na osobním počítači.

AT příkaz – Příkaz začínající znaky AT aloužící pro nastavování a zjišťování stavu.

Escape seqence – Sequence znaků sloužících u bezdrátových modulů pro výstup ze základního stavu do režimu AT příkazů.

Přílohy

ASCII kódovací tabulka

Pro příklad, ASCII číslo pro písmeno "A" je 65 decimalně nebo $41 v hexadecimal.

Znak	Číslo	Znak	Číslo	Znak	Číslo	Znak	Číslo
NUL(^@)	0$00	DLE(^P)	16$10	SPACE	32$20	0	48$30
SOH(^A)	1$01	DC1(^Q)	17$11	!	33$21	1	49$31
STX(^B)	2$02	DC2(^R)	18$12	"	34$22	2	50$32
ETX(^C)	3$03	DC3(^S)	19$13	#	35$23	3	51$33
EOT(^D)	4$04	DC4(^T)	20$14	$	36$24	4	52$34
ENQ(^E)	5$05	NAK(^U)	21$15	%	37$25	5	53$35
ACK(^F)	6$06	SYN(^V)	22$16	&	38$26	6	54$36
BEL(^G)	7$07	ETB(^W)	23$17	'	39$27	7	55$37
BS(^H)	8$08	CAN(^X)	24$18	(	40$28	8	56$38
HT(^I)	9$09	EM(^Y)	25$19	)	41$29	9	57$39
LF(^J)	10$0A	SUB(^Z)	26$1A	*	42$2A	:	58$3A
VT(^K)	11$0B	ESC(^[)	27$1B	+	43$2B	;	59$3B
FF(^L)	12$0C	FS(^\)	28$1C	,	44$2C	<	60$3C
CR(^M)	13$0D	GS(^])	29$1D	-	45$2D	=	61$3D
SO(^N)	14$0E	RS(^^)	30$1E	.	46$2E	>	62$3E
SI(^O)	15$0F	US(^_)	31$1F	/	47$2F	?	63$3F

Znak	Číslo	Znak	Číslo	Znak	Číslo	Znak	Číslo
@	64$40	P	80$50	`	96$60	p	112$70
A	65$41	Q	81$51	a	97$61	q	113$71
B	66$42	R	82$52	b	98$62	r	114$72
C	67$43	S	83$53	c	99$63	s	115$73
D	68$44	T	84$54	d	100$64	t	116$74
E	69$45	U	85$55	e	101$65	u	117$75
F	70$46	V	86$56	f	102$66	v	118$76
G	71$47	W	87$57	g	103$67	w	119$77
H	72$48	X	88$58	h	104$68	x	120$78
I	73$49	Y	89$59	i	105$69	y	121$79
J	74$4A	Z	90$5A	j	106$6A	z	122$7A
K	75$4B	[	91$5B	k	107$6B	{	123$7B
L	76$4C	\	92$5C	l	108$6C	|	124$7C
M	77$4D	]	93$5D	m	109$6D	}	125$7D
N	78$4E	^	94$5E	n	110$6E	~	126$7E
O	79$4F	_	95$5F	o	111$6F	DEL	127$7F

JAVA kódy specielních znaků

Pro vyjádření některých speciálních znaků v JAVĚ lze použít tyto sekvence:

\b	backspace
\t	tabulátor
\n	nový řádek (LF)
\f	nová stránka (FF)
\r	návrat vozíku (CR)
\"	uvozovky

\'	apostrof
\\	zpětné lomítko
\xxx	znak zapsaný v osmičkovém kódu (000-0377)
\uxxxx	znak Unicode zapsaný hexadecimálně

Modul X-Bee ve zkušební desce