STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ

Vojtěch Boček

Brno 2012

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor SOČ: 18. Informatika

GRAFICKÉ UŽIVATELSKÉ ROZHRANÍ

Autor: **Vojtěch Boček**

Škola: **SPŠ a VOŠ technická,**
Sokolská 1 602 00 Brno

Konzultant: **Jakub Streit**

Brno 2012

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně, použil jsem pouze podklady (literaturu, SW atd.) citované v práci a uvedené v přiloženém seznamu a postup při zpracování práce je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Brně dne: 6.3.2012 podpis:

Poděkování

Děkuji Jakubu Streitovi za rady, obětavou pomoc, velkou trpělivost a podnětné připomínky poskytované během práce na tomto projektu, a Martinu Vejnárovi za informace o jeho programátoru Shupito.
Dále děkuji organizaci DDM Junior, za poskytnutí podpory.
Také bych chtěl poděkovat panu profesorovi Mgr. Miroslavu Burdovi za všeobecnou pomoc s prací.
V neposlední řadě děkuji Martinu Foučkovi za rady a pomoc s Qt Frameworkem.

Tato práce byla vypracována za finanční podpory JMK.

Anotace

Cílem této práce bylo vytvořit uživatelské prostředí určené k parsování a zobrazování surových dat posílaných z mikrokontrolérů v robotech, digitálních sondách apod. Hlavní vlastností programu je modulárnost – rozdělení na podčásti určené ke specifickým úkonům (Terminál, grafický parser, vykreslování grafů).

Klíčová slova: parser, analýza dat, program

Annotation

Purpose of this labor is to create graphical user interface for parsing and displaying raw data sent from embedded devices, robots, digital probes and other devices which are using microcontrollers. Main feature of this application is modularity – it is divided to sub-sections designed for specific operations (Terminal, graphical parser, graph drawer).

Key words: parser, data analysis, program

Obsah

Úvod

Při stavbě robotů (například na soutěž Eurobot) jsem se setkal s problémem při zpracovávání dat z poměrně velkého množství senzorů, které robot obsahuje, a jejich přehledného zobrazování. Nenašel jsem žádný program, který by mi vyhovoval – k dispozici jsou pouze komerční aplikace, které stojí poměrně velké množství peněz, anebo aplikace které dokáží zobrazovat pouze v jednom formátu – typicky graf. Z tohoto důvodu jsem se rozhodl napsat vlastní program.

Existující programy

Aplikace s tímto zaměřením jsem našel pouze několik málo aplikací.

- **SerialChart**[1] je open-source program[2] pro parsování a zobrazování dat přicházející ze sériového portu. Je jednoduchý a přehledný, dokáže však zobrazovat pouze graf a nastavení je třeba ručně napsat.
- **WinWedge**[3] je komerční program který dokáže zpracovávat data přicházející sériovým portem a zobrazovat je jako graf v MS Excel nebo ve webové stránce. Dokáže také posílat příkazy zpět do zařízení, oproti Lorris má však horší ovládání (hlavně kvůli nutnosti použít další program pro zobrazování), má užší možnosti použití a je dostupný pouze pro MS Windows.
- **StampPlot Pro**[4] dokáže zobrazovat příchozí data ve widgetech zvolených uživatelem, podobně jako modul Analyzér v Lorris, má však komplikované ovládání, nemá otevřený zdrojový kód, je dostupný pouze pro MS Windows a pod MS Windows 7 nefunguje.

[1] *Analyse and chart serial data from RS-232 COM ports* – http://code.google.com/p/serialchart/

[2] Program s otevřeným zdrojovým kódem

[3] *RS232 data collection software* – http://www.taltech.com/products/winwedge/

[4] *Graphical Data Acquisition and Control* – http://www.selmaware.com/stampplot/index.htm

1 Popis rozhraní

Svůj program jsem pojmenoval "Lorris", je vytvořený v C++ a využívá Qt Framework[5] (v4.7), což je multiplatformní framework, který mimo jiné umožňuje spustit aplikaci na více systémech – testoval jsem na Debian Linux[6] (Wheezy, 64bit) a Windows 7.

1.1 Web a repozitář programu

GIT[7] repozitář programu jsem vytvořil na serveru GitHub[8], který kromě hostingu repozitáře poskytuje i několik dalších služeb, mezi nimi i hosting webu projektu. Na webu, který jsem vytvořil, jsou odkazy ke stažení spustitelných souborů pro Windows, popis programu, video s představením programu (6 min.), ukázky z programu (screenshoty) a návod ke zkompilování pro MS Windows a Linux.

- Repozitář: https://github.com/Tasssadar/Lorris
- Web (česká verze):
  http://tasssadar.github.com/Lorris/cz/index.html
- Web (anglická verze):
  http://tasssadar.github.com/Lorris/index.html

V repozitáři nadále probíhá aktivní vývoj.

1.2 Struktura aplikace

Program je navrhnutý jako modulární aplikace, aby mohl zastřešit několik samostatných částí, které však mají podobnou oblast použití. Základní část

[5] *Cross–platform application and UI framework* – http://qt.nokia.com/
[6] *The Universal Operating System* – http://www.debian.org/
[7] *GIT* – distribuovaný systém správy verzí
[8] *GitHub – Social Coding* – https://github.com

programu poskytuje připojení k zařízení (např. robot, deska s čipem) a ukládání nastavení aplikace, samotné zpracování dat probíhá v modulech, které jsou otevírány v panelech – podobně jako stránky ve webovém prohlížeči.

Možnosti připojení k zařízení:

- Sériový port
- Shupito Tunel (virtuální sériový port, viz kapitola 4.1)
- TCP socket[9]
- Načtení dat ze souboru

Je možné mít připojeno více různých modulů na jedno zařízení.

[9] *Transmission Control Protocol* – připojení přes internet.

2 Modul: Analyzér

Obrázek 1: Modul analyzér

Tento modul parsuje data (strukturované do packetů) přicházející ze zařízení a zobrazuje je v grafických "widgetech". Zpracovaná data si aplikace ukládá do paměti – listování packety je možné pomocí posuvníku a boxu v horní části okna. Data (přijatá data, struktura packetů a rozestavení a nastavení widgetů) je také možné uložit do souboru a později zase v programu otevřít.

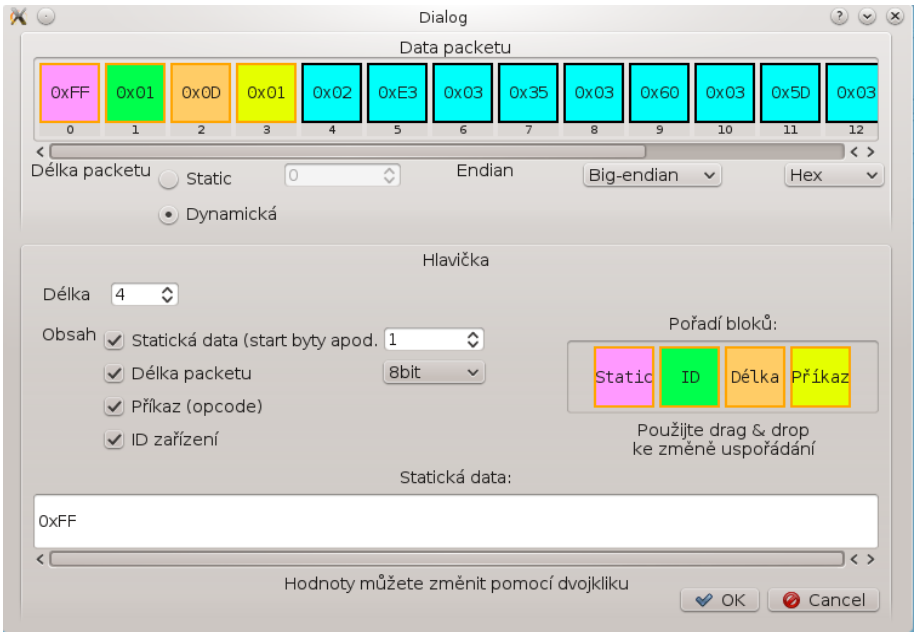
Struktura dat se nastavuje v samostatném dialogu (viz obrázek 2), kde je možno nastavit délku packetu, jeho endianness[10], přitomnost hlavičky a její obsah – statická data ("start byte"), délka packetu(pokud je proměnná), příkaz a ID zařízení. Podle příkazu a ID zařízení je možno později data filtrovat.

[10] *Endianness* – pořadí uložení bytů v paměti počítače

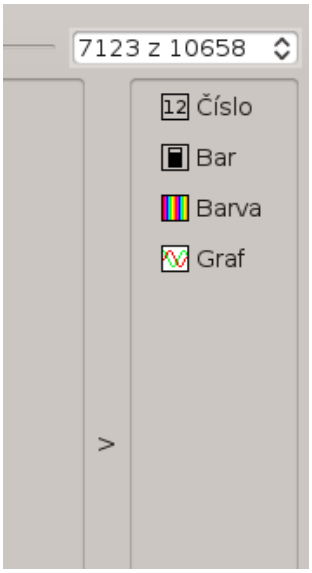
Po nastavení struktury se přijatá data začnou po packetech zobrazovat v horní části okna, a v pravé části se zobrazí sloupeček s dostupnými zobrazovacími widgety. Widgety se dají pomocí drag&drop principu „vytahat“ na plochu v prostřední části okna. Data se k widgetu přiřadí taktéž pomocí drag&drop, tentokrát přetažení prvního bytu dat na widget.

Poté widget zobrazuje data tohoto bytu, nebo tento byte bere jako první, pokud jsou data delší. Aby bylo možné zpětně poznat, který byte je k widgetu přiřazen, je po najetí myši na widget červeně zvýrazněn.

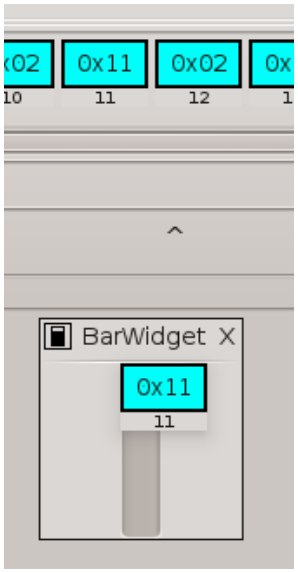
Nastavení widgetu jsou přistupná v kontextovém menu po pravém kliknutí myší na widget. Nastavit lze jméno a další parametry podle typu widgetu – podrobněji jsou možnosti nastavení popsány u jednotlivých widgetů. Widgety je taktéž možné „uzamknout“, aby nebylo možné je zavřít, měnit jejich pozici a velikost.



Obrázek 2: Dialog nastavení struktury dat



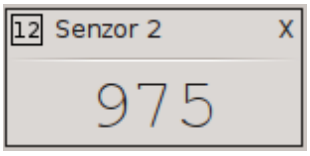
(a) Seznam widgetů



(b) Přiřazení dat pomocí drag&drop

Obrázek 3: Widgety

2.1 Widget: číslo



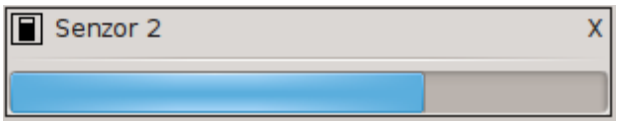
Tento widget dokáže zobrazovat celá čísla (se znaménkem i bez, 8 až 64 bitů dlouhé) a desetinná čisla (single-precision[11], 32bit a 64bit).
Widget dále dokáže zarovnat číslo na maximální délku jeho datového typu a formátovat ho těmito způsoby:

- Desítkový – číslo v desítkové soustavě

[11]Standartní formát uložení desetinných čísel v jazyku C a dalších (standart IEEE 754-2008).

- Desítkový s exponentem – použije exponent pro zapsaní velkých čísel. Dostupné pouze pro desetinná čísla.
- Hexadecimální – výpis v šestnáctkové soustavě. Dostupné pouze pro přirozená čísla.
- Binární – zobrazí číslo ve dvojkové soustavě. Dostupné pouze pro přirozená čísla.

2.2 Widget: sloupcový bar

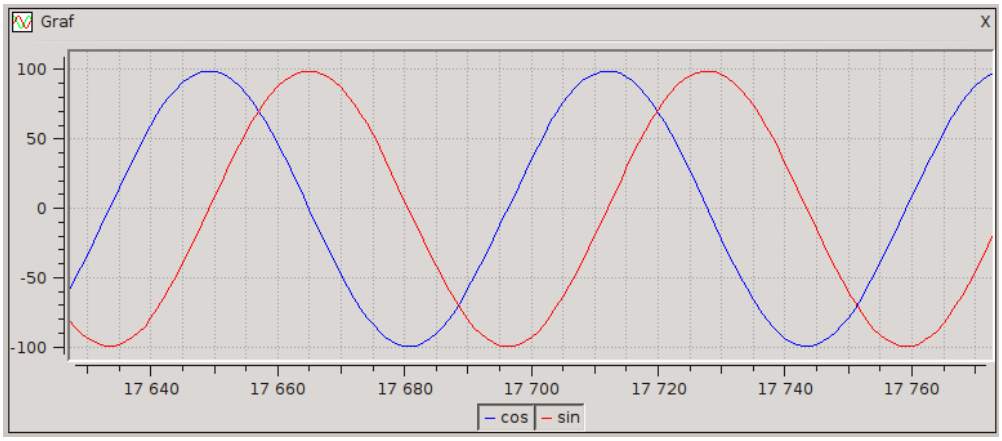


Widget zobrazuje hodnotu ve sloupcovém baru. Lze nastavit datový typ vstupních dat (stejně jako u čísla), orientaci (vertikální nebo horizontální) a rozmezí zobrazovaných hodnot.

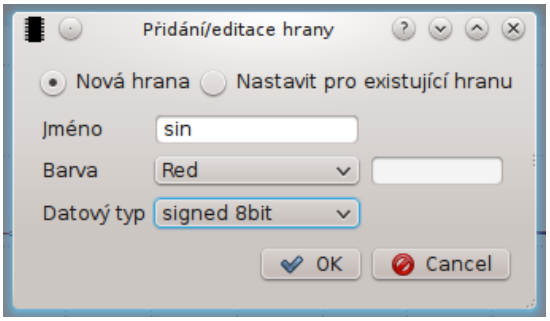
2.3 Widget: barva

Tento widget ukáže 24-bitové hodnoty RGB jako barevný obdélník. Dokáže provést korekci jasu všech barev nebo každé z barev RGB zvlášť.

2.4 Widget: graf

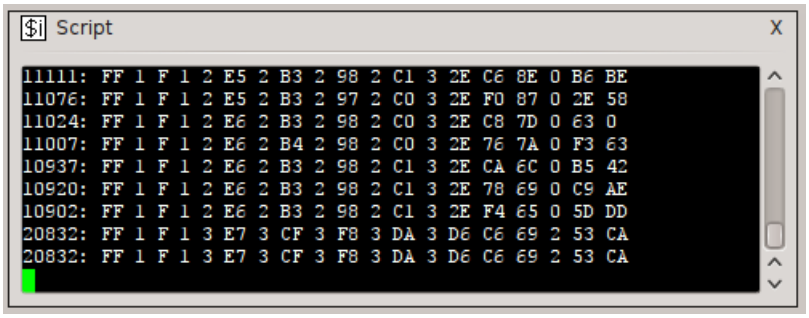


Widget graf zobrazuje hodnoty v grafu – na osu x se vynáší pořadí dat a na osu y hodnoty dat. Lze nastavovat jméno, barvu a datový typ křivky grafu, automatické posouvání grafu, velikost vzorku, měřítko os grafu a zobrazení legendy. Kliknutí na křivku grafu v legendě tuto křivku skryje. Měřítko osy se ovládá otáčením kolečka myši po najetí kurzoru nad osu, po najetí do prostoru grafu se podobně ovládá měřítko celého grafu.



Obrázek 4: Dialog pro nastavení parametrů křivky grafu

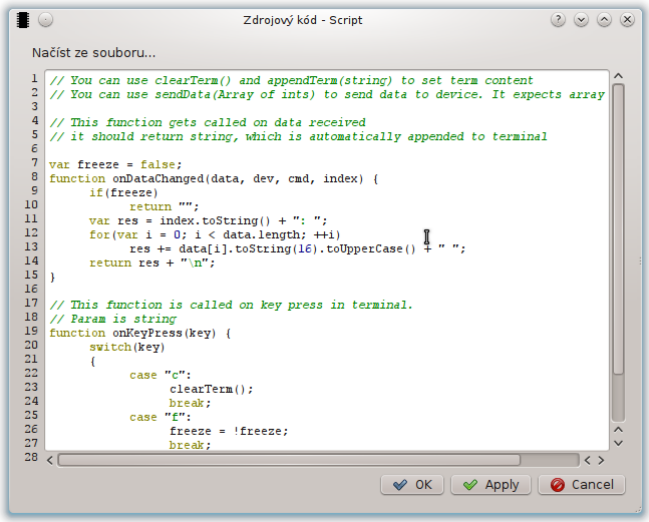
2.5 Widget: script



Tento widget umožňuje zpracovávání dat pomocí scriptu, který si napíše sám uživatel. Jazyk, ve kterém se tento script píše je QtScript (jazyk založený na standartu ECMAScript[12], stejně jako JavaScript[13], díky tomu jsou tyto jazyky velmi podobné). Script může zpracovávat příchozí data, reagovat na stisky kláves a posílat data do zařízení. Základní výstup může být zobrazen v terminálu (viz obrázek nad tímto textem), je však možné využít ke zobrazování také ostatní widgety (číslo, bar, ...) - script si je vytvoří jako objekt a nastavuje do nich data. Reference k vestavěným funkcím, které lze použít ve scriptu je ve wiki u repozitáře programu: https://github.com/Tasssadar/Lorris/wiki.

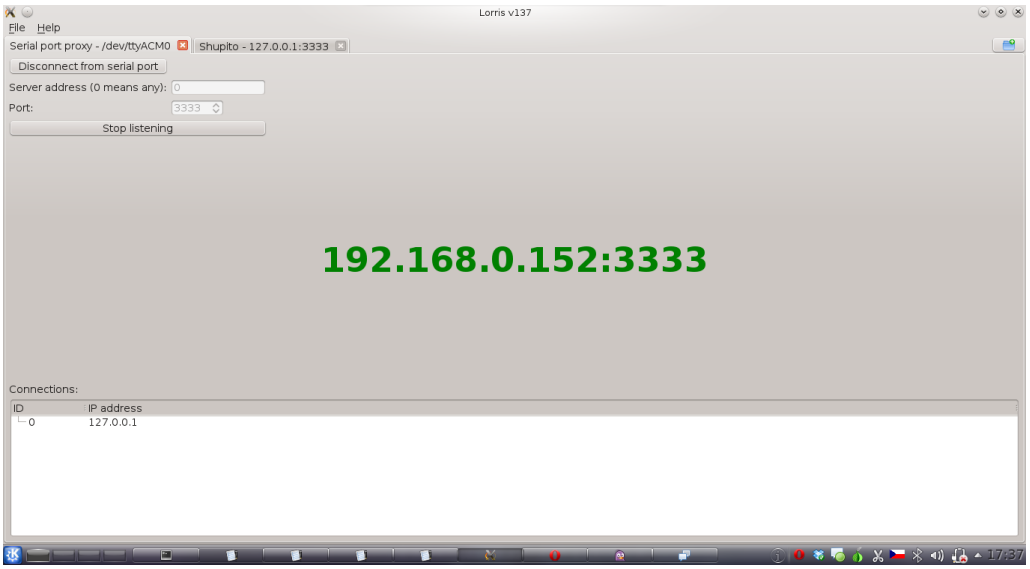
[12] *ECMAScript* – scriptovací jazyk stadartu ECMA-262 a ISO/IEC 16262

[13] *JavaScript* – objektově orientovaný skriptovací jazyk, používaný hlavně na webu



Obrázek 5: Dialog pro nastavení zdrojového scriptu

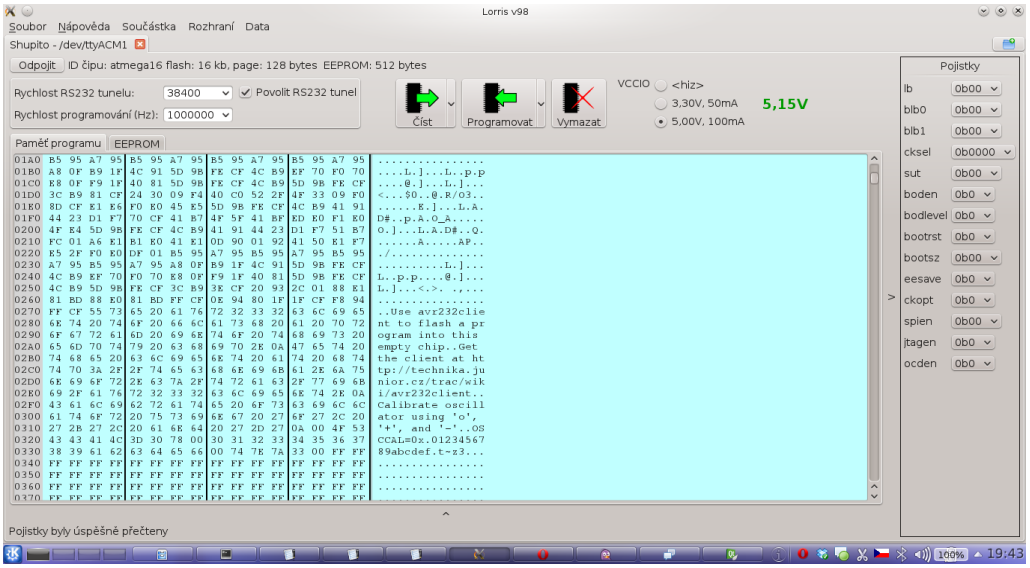
3 Modul: Proxy mezi sériovým portem a TCP socketem



Obrázek 6: Proxy mezi sériovým portem a TCP socketem

Jednoduchá proxy mezi sériovým portem a TCP socketem. Vytvoří server, na který je možné se připojit z Lorris nebo jiného programu na jiném počítači. Po připojení se přeposílají data ze sériového portu připojeným klientům a naopak.

4 Modul: Shupito



Obrázek 7: Modul Shupito

Shupito je programátor mikročipů vytvořený Martinem Vejnárem, který dokáže programovat mikrokontroléry pomocí ISP[14], PDI[15] a JTAG[16] rozhraní.

Modul v mojí práci dokáže obsluhovat programátor Shupito – nastavovat výstupní napětí, číst a programovat paměť čipů (flash i EEPROM) a číst a měnit pojistky. Jako výstupní i vstupní data používá soubory ve formátu Intel HEX32[17]. Způsob komunikace s programátorem je přenesen z oficiálního ovládacího programu[3], který je však na rozdíl od Lorris dostupný pouze pro MS Windows.

[14] *In-system programming* – rozhraní, které umožňuje programovat čipy bez dalšího zařízení přímo v desce plošného spoje.

[15] *Program and Debug Interface* – rozhraní firmy Atmel umožňující programování čipů přímo na desce, podobně jako ISP

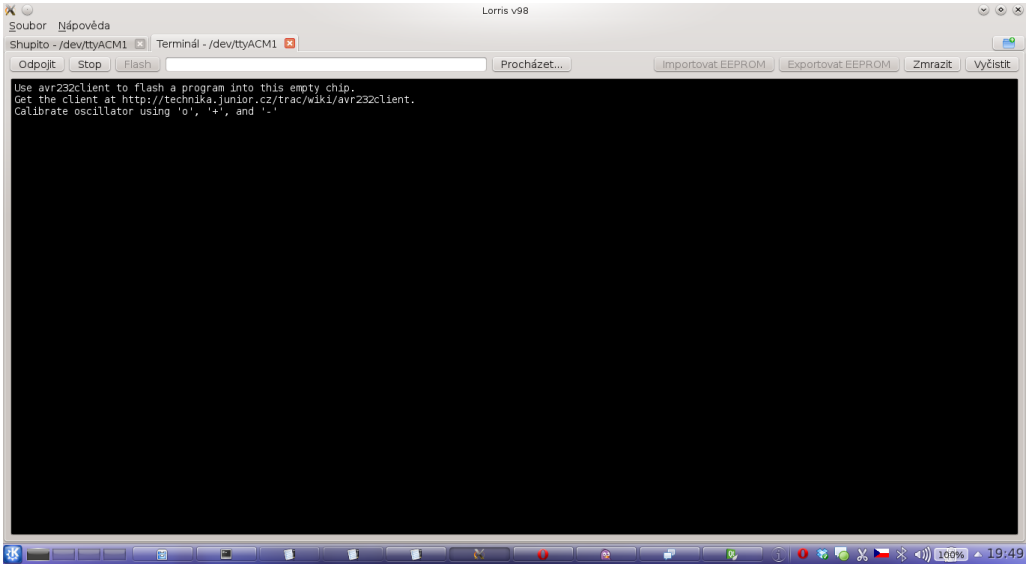
[16] *Joint Test Action Group* – rozhraní podle standartu IEEE 1149.1 umožňující mimo jiné programování a debugování čipů

[17] *Intel HEX32* – formát souborů obsahující paměť čipu

4.1 RS232 tunel

Shupito dokáže vytvořit tunel[18] pro RS232 linku z programovaného čipu do počítače. Lorris umí této funkce využít – aktivní tunel se zobrazí jako další typ připojení a je možné se na něj připojit v ostatních modulech.

5 Modul: Terminál



Obrázek 8: Modul terminál

Klasický terminál – zobrazuje data přijatá přes sériový port a posílá stisky kláves. Je obohacený o podporu bootloaderu pro mikrokontroléry AVR ATmega (bootloader byl taktéž napsaný Martinem Vejnarem), který umožňuje jejich programování přes RS232 linku. Informace o protokolu bootloaderu jsem získal z oficiálního programu určeného k programování přes tento bootloader, avr232client.

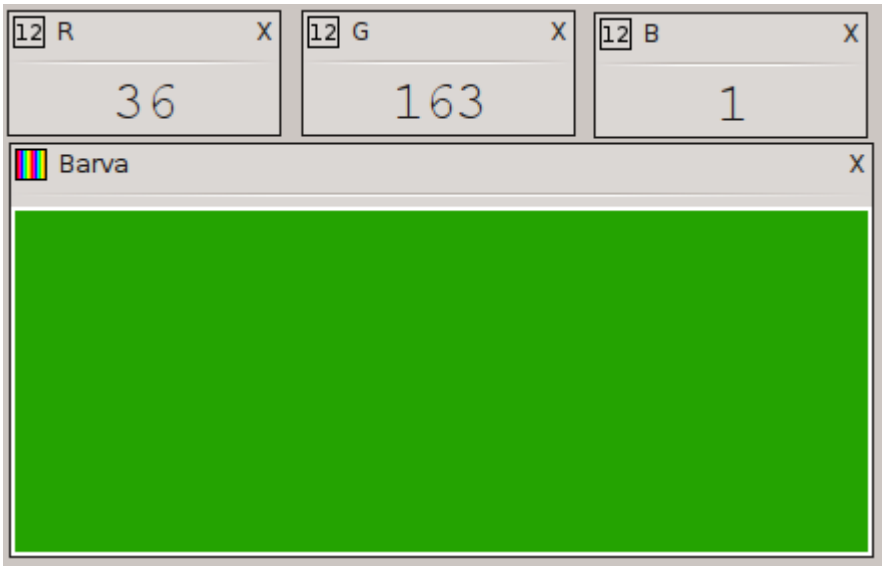
[18] Přímé spojení programovaného čipu a počítače přes programátor.

6 Příklady použití

6.1 Testování barevného senzoru

Situace: Stavím robota do soutěže (Eurobot, RobotChallange, ...), ve které je možné se na herním poli orientovat podle barvy. Chci barevný senzor otestovat, proto jsem na nepájivém poli postavil jednoduchý obvod s čipem, na který je senzor připojený. Čip bude dávat senzoru pokyny k měření a vyčítat z něj RGB hodnoty, které následně pošle do RS232 linky.

Řešení: Program, který bude ze senzoru číst hodnoty naprogramuji do čipu pomocí programátoru Shupito, který také poskytne tunel pro RS232 linku. Na tento tunel se připojím modulem Analyzér, ve kterém díky widgetu "barva" mohu vidět barvu, kterou senzor rozpoznal.



Obrázek 9: Barva v modulu Analyzér

6.2 Testování enkodérů

Situace: Potřebuji otestovat přesnost magnetických enkodérů, které však odesílají data o úhlu natočení rozdělené v několika bytech v takovém formátu, který znemožňuje použití např. terminálu.

Řešení: Nechci za tímto účelem stavět a programovat novou desku s dalším mikročipem, připojím tedy enkodér k počítači. V Lorris otevřu modul analyzér a ve widgetu "script" napíši jednoduchý script, který složí úhel do jednoho čísla a zobrazí ho ve widgetu "číslo".

6.3 Ladění PID regulátoru

Situace: Robot kvůli rozdílnému výkonu motorů nejede rovně. Tento problém jsem se rozhodl řešit pomocí PID regulátoru, pro jehož správnou funkci je potřeba nastavit několik konstant.

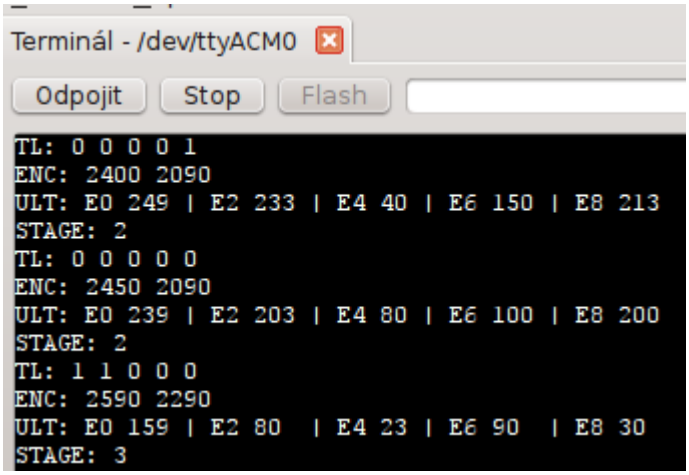
Řešení: Program v robotovi mi posílá aktualní výkon motorů a nastavení konstant PID regulátoru a umožňuje přenastavení těchto konstant a ovládání robota. Tento program do robota nahrávám přes bluetooth pomocí modulu Terminál, protože čip má v sobě bootloader – díky tomu nemusím mít připojený programátor.

V modulu analyzér si zobrazím aktuální hodnoty PID regulátoru (jako číslo) a výkon motorů (jako graf či číslo). Do widgetu script napíšu jednoduchý script, který po stisku kláves změní nastavení konstant regulátoru nebo rozjede/zastaví robota.

6.4 Vývoj robota pro soutěž Eurobot 2011

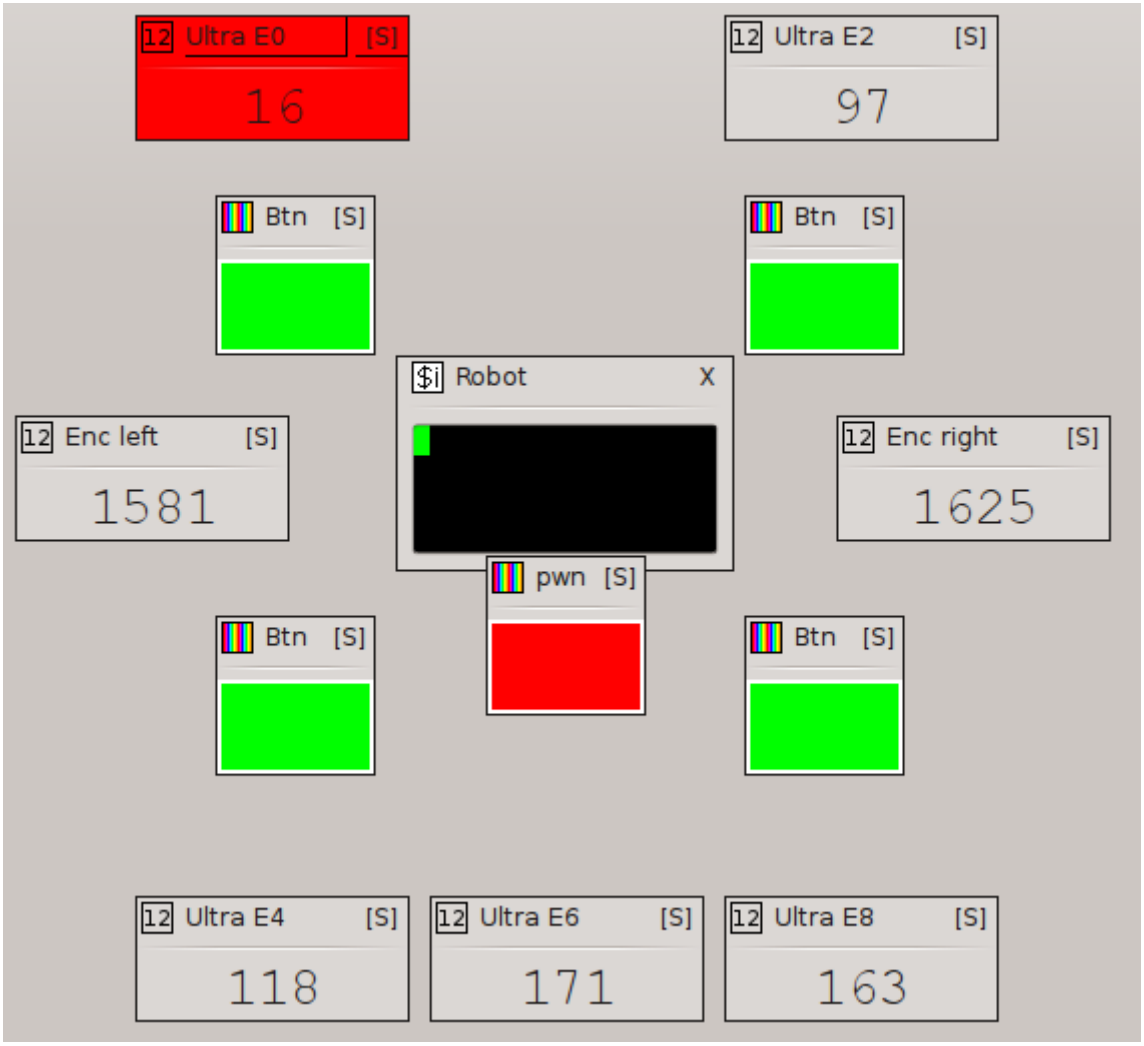
V minulém roce jsem se zúčastnil soutěže Eurobot. Cíl soutěže je každý rok jiný, v minulém ročníku bylo cílem hrát něco jako zjednodušené šachy. Herní hřiště bylo rozděleno na barevnou šachovnici a leželi na něm "pěšci" (žluté disky), které měli roboti posouvat na políčka svojí barvy, případně z nich stavět věže. Vyhrával robot s největším počtem bodů, které získával za pěšce na polích svojí barvy a postavené věže. Roboti navíc musí mít vyřešenou detekci soupeře, aby do sebe nenaráželi (např. pomocí ultrazvukových měřičů vzdálenosti).

Robot našeho týmu byl poměrně jednoduchý, přesto však obsahoval 5 ultrazvukových měřáků vzdálenosti, dva enkodéry a 5 tlačítek (detekce nárazu na mantinel a pěšce, kterého mohl robot převážet). Tyto senzory produkují poměrně značné množství dat, které se v terminálu zobrazuje nepřehledně.



Obrázek 10: Data z robota v terminálu

Zkušenost s programováním a laděním robota byla jedním z hlavních důvodů pro výběr zvoleného tématu práce. S použitím programu Lorris vypadá výstup následnovně:



Obrázek 11: Data z robota v analyzéru

Všechny widgety jsou rozmístěné stejně jako na robotovi – 2 ultrazvuky měří vzdálenost vepředu, 3 vzadu; tlačítka pro detekci mantinelu jsou na každém rohu, tlačítko uvnitř robota ověřuje, zda robot nabral pěšce a enkodéry na obou kolech měří ujetou vzdálenost.

Widget "Robot" uprostřed reprezentuje tělo robota. Widgety *číslo* s názvy "Ultra E0…8" ukazují vzdálenost z ultrazvukových měřáků. Ve widgetu "Ultra E0" je vzdálenost menší než 25 cm, což je hraniční vzdálenost, po

které se robot zastaví, aby nevrazil do soupeře – aby widget na tuto skutečnost upozornil, má červené pozadí.

Widgety *barva* s názvy "btn" jsou tlačítka značící náraz do mantinelu a "pwn" je tlačítko které se stiskne pokud je v robotovi pěšec. Tlačítko které má zelenou barvu není stisklé, tlačítko s červenou barvou stisklé je.

Poslední widgety *číslo* s názvy "Enc left" a "Enc right" vypisují ujetou vzdálenost z enkodérů pravého a levého kola.

7 Knihovny třetích stran, licence

7.1 Knihovny třetích stran

Qwt[19] je knihovna pro Qt Framework obsahující tzv. widgety pro aplikace technického charakteru – grafy, sloucové ukazatele, kompasy a podobně. Ve svojí práci zatím z této knihovny používám pouze graf (v modulu analyzéru).

QExtSerialPort[20] poskytuje připojení k sériovému portu a také dokáže vypsat seznam nalezených portů v počítačí.

QHexEdit2[21] je hex editor použitý v modulu programátoru Shupito na zobrazování obsahu paměti. V této knihovně jsem upravoval několik málo drobností, týkajících se především vzhledu.

[19] *Qt Widgets for Technical Applications* – http://qwt.sourceforge.net/

[20] *Qt interface class for old fashioned serial ports* – http://code.google.com/p/qextserialport/

[21] *Binary Editor for Qt* – http://code.google.com/p/qhexedit2/

7.2 Licence

Lorris je dostupný pod licencí GNU GPLv3[22], licence použitých programů a knihoven jsou následující:

- **Qt Framework** je distribuován pod licencí GNU LGPLv2.1[23]
- **Qwt** je distribuováno pod Qwt license[24], která je založená na GNU LGPLv2.1
- **QExtSerialPort** je distribuován pod The New BSD License[25]
- **QHexEdit2** je distribuován pod licencí GNU LGPLv2.1
- **avr232client** je distribuován pod licencí Boost Software License v1.0[26]

[22] *GNU General Public License* – http://gplv3.fsf.org/

[23] *GNU Lesser General Public License* – http://www.gnu.org/licenses/lgpl-2.1.html

[24] *Qwt license* – http://qwt.sourceforge.net/qwtlicense.html

[25] *The New BSD License* – http://www.opensource.org/licenses/bsd-license.php

[26] *Boost Software License* – http://www.boost.org/users/license.html

Reference

[1] *GIT repozitář Lorris*
https://github.com/Tasssadar/Lorris

[2] *Shupito – Programátor*
http://shupito.net/ (Stav ke dni 28.1.2012)

[3] *avr232client*
http://technika.junior.cz/trac/wiki/avr232client
(Stav ke dni 28.1.2012)