

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 příloženém 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, Martinu Vejnárovi za informace o programátoru Shupito, panu profesorovi Mgr. Miroslavu Burdovi za velkou pomoc s prací a v neposlední řadě Martinu Foučkovi za rady a pomoc při práci s Qt Frameworkem. Dále děkuji organizaci DDM Junior za poskytnutí podpory.

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	8
Požadavky na aplikaci	8
Existující programy	8
Porovnání aplikací	9
1 Popis rozhraní	10
1.1 Web a repozitář programu	10
1.2 Struktura aplikace	10
2 Modul: Analyzér	12
2.1 Widget: číslo	14
2.2 Widget: sloupcový bar	15
2.3 Widget: barva	16
2.4 Widget: graf	17
2.5 Widget: script	18
3 Modul: Proxy mezi sériovým portem a TCP socketem	20
4 Modul: Shupito	21
4.1 RS232 tunel	22
5 Modul: Terminál	22
6 Příklady použití	24
6.1 Testování barevného senzoru	24
6.2 Testování enkodérů	25
6.3 Ladění PID regulátoru	26
6.4 Vývoj robota pro soutěž Eurobot 2011	27
7 Knihovny třetích stran, licence	32
7.1 Knihovny třetích stran	32
7.2 Licence	32
Závěr	33
PŘÍLOHA A: Podrobný popis widgetu <i>script</i> z modulu Analyzér	
lyzér	35
Pár věcí na které je třeba myslet	35

Základní script	36
Dostupné funkce	36
Vytvoření widgetu	37
Dostupné funkce widgetů	39
Widget číslo	40
Widget sloupcový bar	40
Widget barva	40
Widget graf	41
Widget vstup	42
PŘÍLOHA B: Reference	45
PŘÍLOHA C: Seznam obrázků	48

Úvod

Při stavbě robotů na robotické soutěže jsem se setkal s problémem zpracovávání dat z poměrně velkého množství senzorů (několik ultrazvukových měřáků vzdálenosti, enkodéry, které měří ujetou vzdálenost, tlačítka hlídající náraz do mantinelu, ...), které robot obsahuje, a jejich přehledného zobrazování.

Požadavky na aplikaci

Od programu vyžaduji tyto vlastnosti:

1. Možnost zpracovávat data přicházející ze zařízení a přehledně je zobrazovat
2. Podpora co nejvíce formátů příchozích dat
3. Snadné a rychlé používání
4. Možnost běhu i na jiných systémech než je MS Windows
5. Co možná nejnížší cena
6. Snadná rozšiřitelnost (ideálně otevřený zdrojový kód)
7. Nezávislost na další aplikaci (např. MS Office Excel)

Existující programy

Aplikací, které mají podobné určení (tj. vyčítání dat ze sériového portu a jejich zobrazování), jsem našel pouze několik. K dispozici jsou buď komerční aplikace, které stojí poměrně velké množství peněz (a přesto nesplňují všechny požadavky), anebo aplikace, které dokáží zobrazovat data pouze v jednom formátu – typicky graf.

- **SerialChart**[1] je open-source program¹ 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**[2] 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, má však horší ovládání a užší možnosti použití (hlavně kvůli nutnosti použít další program pro zobrazování). Je dostupný pouze pro MS Windows a základní verze stojí \$ 259.
- **Advanced Serial Data Logger**[3] je zaměřený primárně na export dat ze sériové linky do souboru, data dokáže zobrazovat pouze přeposláním do jiné aplikace (např. MS Office Excel), podobně jako WinWedge.
- **StampPlot Pro**[4] dokáže zobrazovat příchozí data ve widgetech zvolených uživatelem, má však komplikované ovládání, nemá otevřený zdrojový kód, je dostupný pouze pro MS Windows a pod verzí 7 nefunguje.

Porovnání aplikací

Následující tabulka shrnuje funkce a vlastnosti jednotlivých programů. Číslování požadavků odpovídá seznamu v kapitole „Požadavky na aplikaci“.

Požadavky:	1	2	3	4	5	6	7
SerialChart	✓	✗	✓	✗	✓	✓	✓
WinWedge	✗	✓	✓	✗	✗	✗	✗
Advanced Serial Data Logger	✗	✓	✓	✗	✗	✗	✗
StampPlot Pro	✓	✓	✗	✗	✓	✗	✓

¹Program s otevřeným zdrojovým kódem

Z těchto důvodů jsem se rozhodl napsat vlastní aplikaci, která bude všechny výše uvedené požadavky splňovat.

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

GIT² repozitář programu jsem vytvořil na serveru GitHub[7], 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

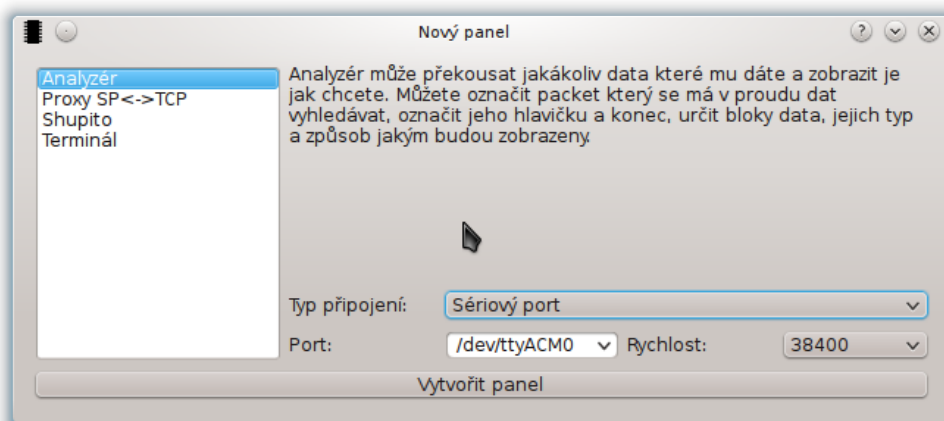
² *GIT* – distribuovaný systém správy verzí

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 socket³
- Načtení dat ze souboru

Je možné mít připojeno více různých modulů na jedno zařízení.



Obrázek 1: Dialog vytvoření panelu

³ *Transmission Control Protocol* – připojení přes internet.

2 Modul: Analyzér



Obrázek 2: 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.

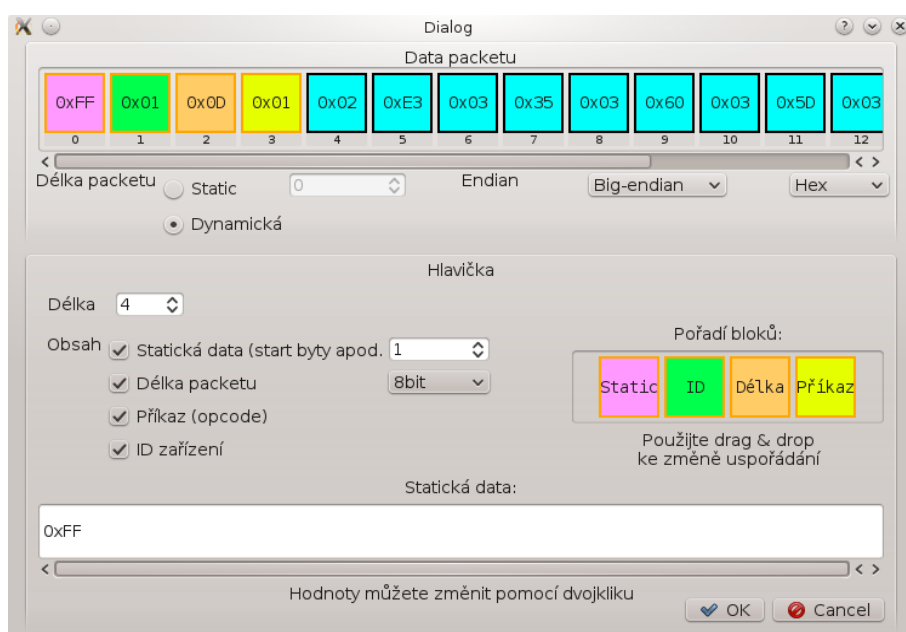
Struktura dat se nastavuje v samostatném dialogu (viz obrázek 3), kde je možno nastavit délku packetu, jeho endianness⁴, přítomnost hlavičky a její obsah – statická data („start byte“), délka packetu (pokud je proměnná), příkaz a ID zařízení. Podle příkazu a ID zařízení je možno později data filtrovat.

⁴*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 „vytáhat“ 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 přístupná v kontextovém menu po pravém kliknutí myši 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 3: Dialog nastavení struktury dat



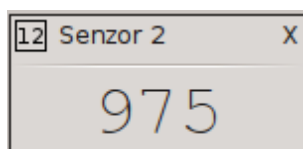
(a) Seznam widgetů



(b) Přiřazení dat pomocí drag&drop

Obrázek 4: Widgety

2.1 Widget: číslo



Obrázek 5: Widget: číslo

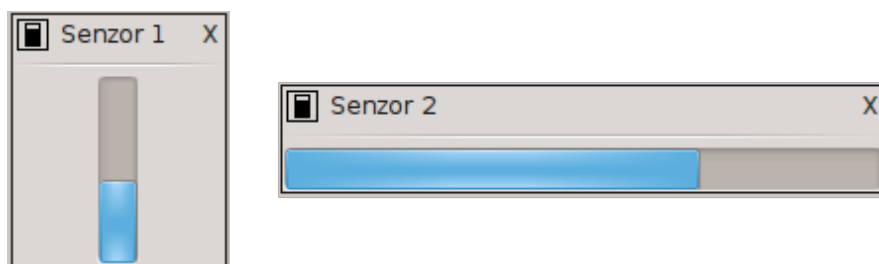
Tento widget dokáže zobrazovat celá čísla (se znaménkem i bez, 8 až 64 bitů dlouhá) a desetinná čísla (single-precision⁵, 32bit a 64bit).

Widget dále dokáže zarovnat číslo na maximální délku jeho datového typu a formátovat ho těmito způsoby:

⁵Standartní formát uložení desetinných čísel v jazyku C a dalších (standart IEEE 754-2008).

- Desítkový – číslo v desítkové soustavě
- Desítkový s exponentem – použije exponent pro zapsání 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.

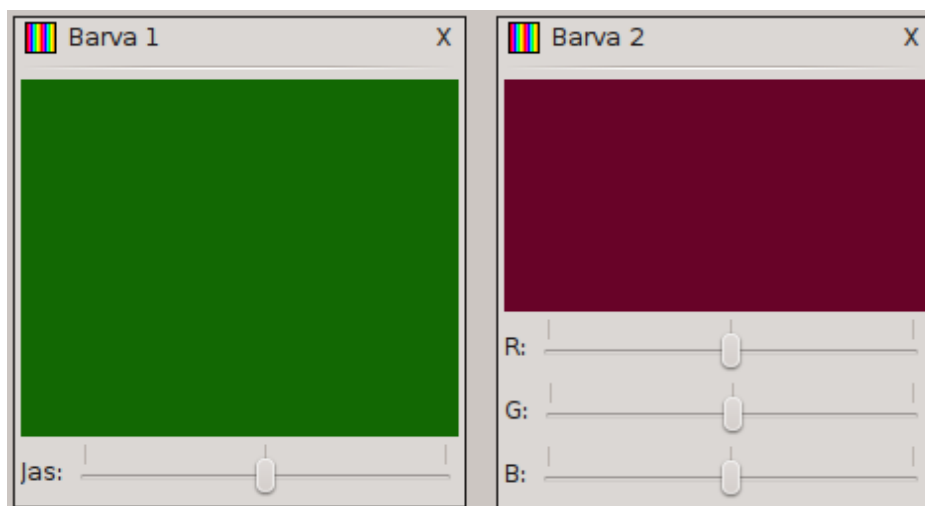
2.2 Widget: sloupcový bar



Obrázek 6: 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.

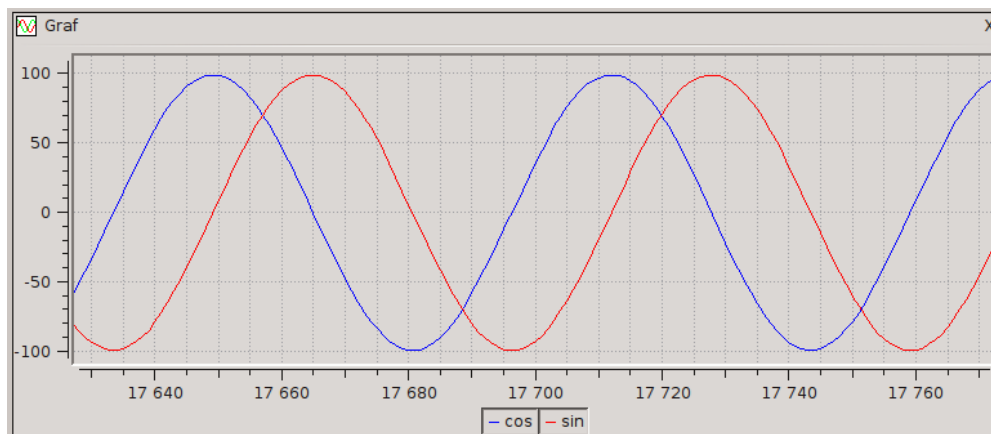
2.3 Widget: barva



Obrázek 7: Widget: barva

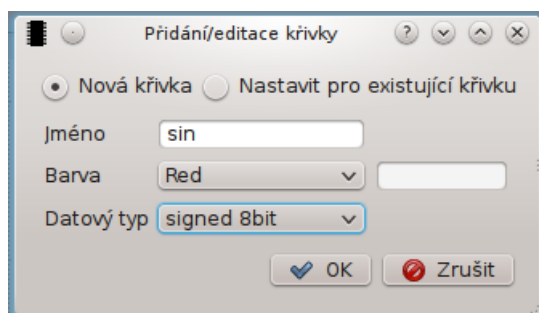
Tento widget ukáže 24-bitové hodnoty RGB jako barevný obdélník. Dokáže provést korekci jasu všech barev nebo každé z barev RGB zvlášť.

2.4 Widget: graf



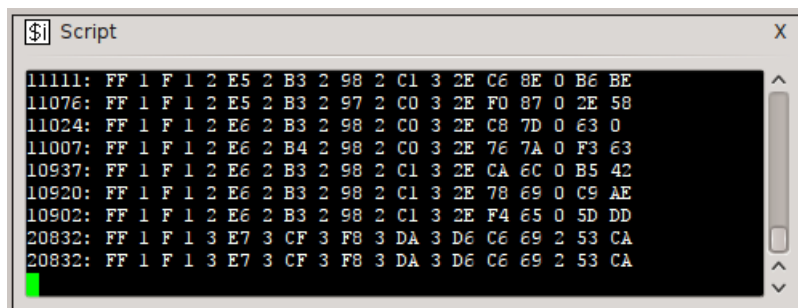
Obrázek 8: Widget: graf

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. Měřítka osy se ovládá otáčením kolečka myši po najetí kurzoru nad osu, po najetí do prostoru grafu se podobně ovládá měřítko celého grafu.



Obrázek 9: Dialog pro nastavení parametrů křivky grafu

2.5 Widget: script

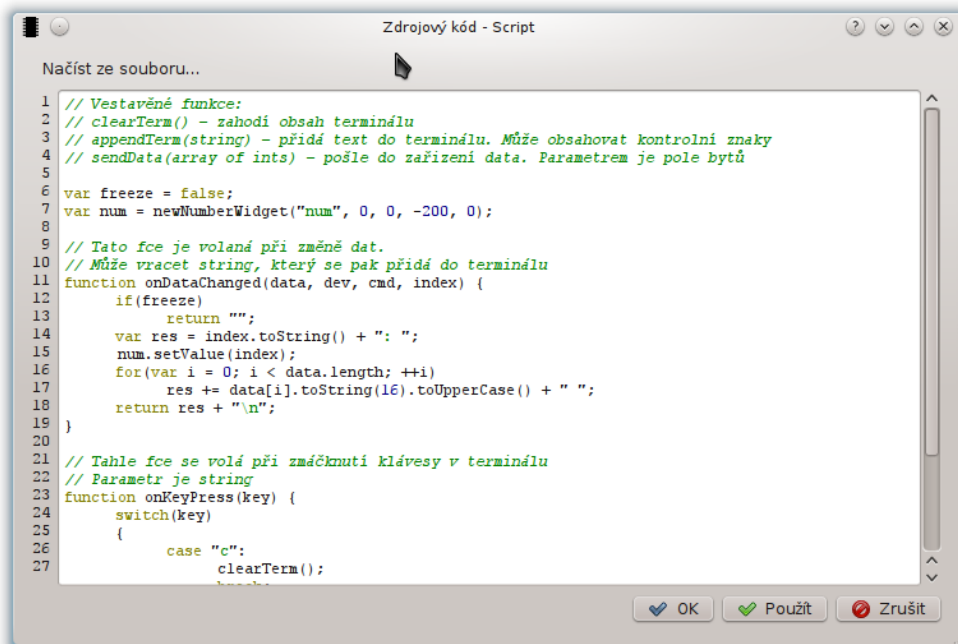


Obrázek 10: Widget: script

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 ECMAScript⁶, stejně jako JavaScript⁷, díky tomu jsou tyto jazyky velmi podobné). Script může zpracovávat přichodí 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 v příloze A.

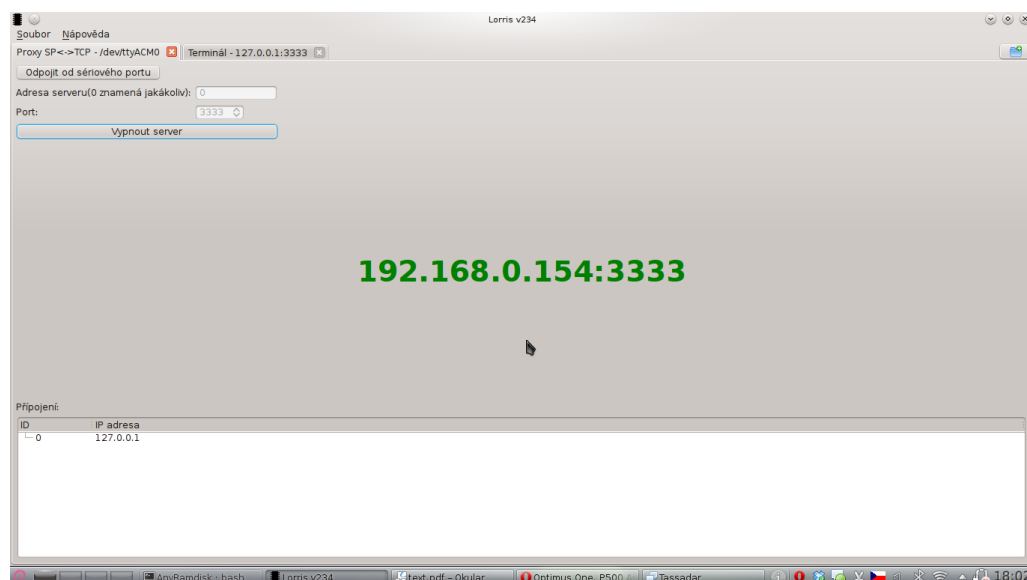
⁶ECMAScript – scriptovací jazyk standartu ECMA-262 a ISO/IEC 16262

⁷JavaScript – objektově orientovaný skriptovací jazyk, používaný hlavně na webu



Obrázek 11: Dialog pro nastavení zdrojového scriptu

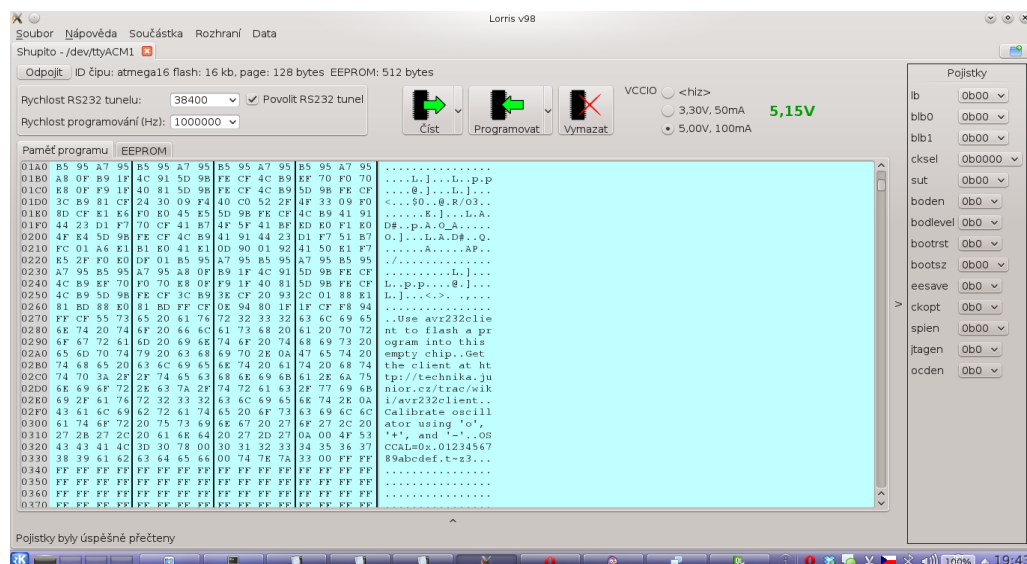
3 Modul: Proxy mezi sériovým portem a TCP socketem



Obrázek 12: 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 13: Modul Shupito

Shupito je programátor mikročipů vytvořený Martinem Vejnárem, který dokáže programovat mikrokontroléry pomocí ISP⁸, PDI⁹ a JTAG¹⁰ 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 HEX32¹¹. Způsob komunikace s programátorem je přenesen z oficiálního ovládacího programu[8], který je však na rozdíl od Lorris dostupný pouze pro MS Windows.

⁸*In-system programming* – rozhraní, které umožňuje programovat čipy bez dalšího zařízení přímo v desce plošného spoje.

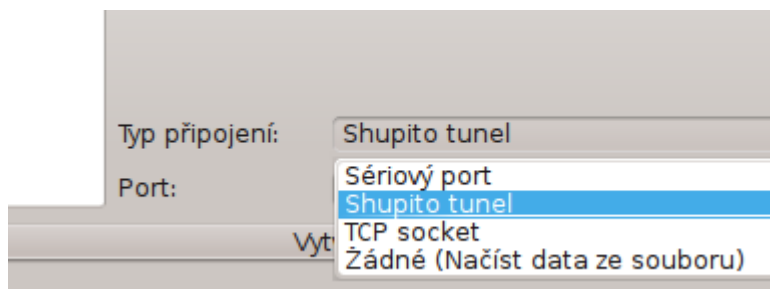
⁹*Program and Debug Interface* – rozhraní firmy Atmel umožňující programování čipů přímo na desce, podobně jako ISP

¹⁰*Joint Test Action Group* – rozhraní podle standartu IEEE 1149.1 umožňující mimo jiné programování a debugování čipů

¹¹*Intel HEX32* – formát souborů obsahující paměť čipu

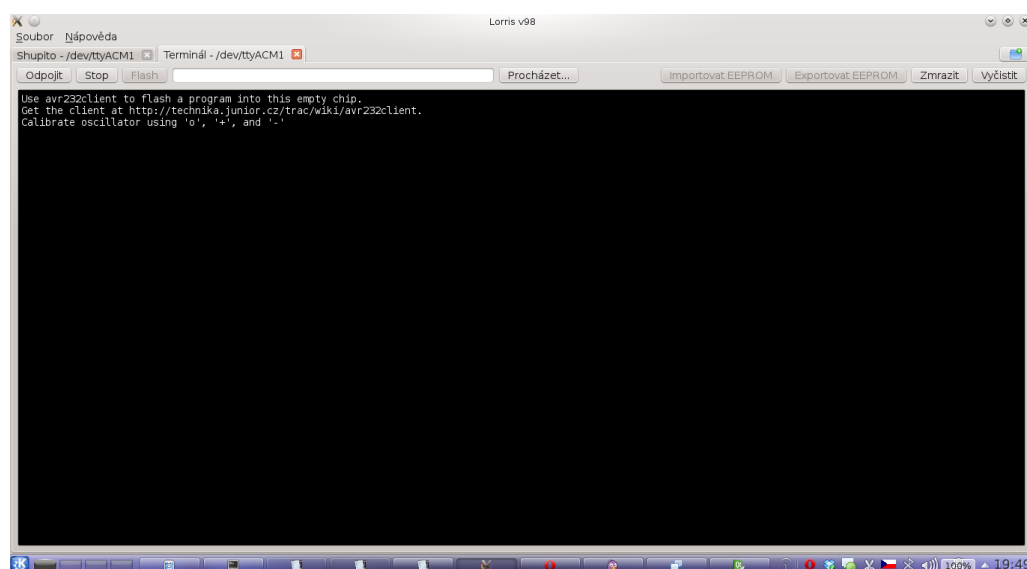
4.1 RS232 tunel

Shupito dokáže vytvořit tunel¹² pro RS232 linku z programovaného čipu do počítače. Lorris umí této funkci využít – aktivní tunel se zobrazí jako další typ připojení a je možné se na něj připojit v ostatních modulech.



Obrázek 14: Možnost Shupito Tunel v dialogu připojení k zařízení

5 Modul: Terminál



Obrázek 15: Modul terminál

¹²Přímé spojení programovaného čipu a počítače přes programátor.

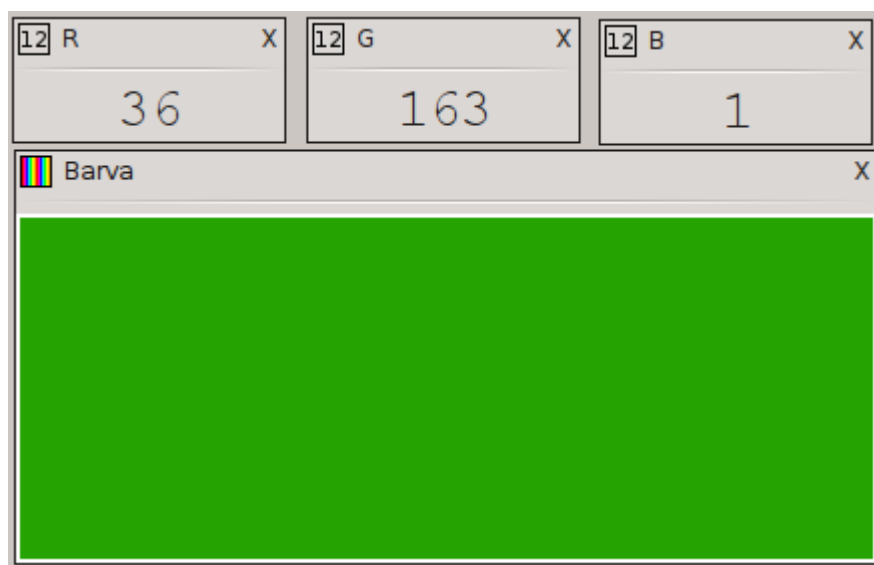
Klasický terminál – zobrazuje data přijatá přes sériový port a posílá stisky kláves. Je obohacený o podporu bootloaderu pro mikrokontroléry AVR ATmega (bootloader byl taktéž napsaný Martinem Vejnarem), který umožňuje jejich programování přes RS232 linku. Informace o protokolu bootloaderu jsem získal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

6 Příklady použití

6.1 Testování barevného senzoru

Situace: Stavím robota do soutěže (Eurobot, RobotChallenge, ...), 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.

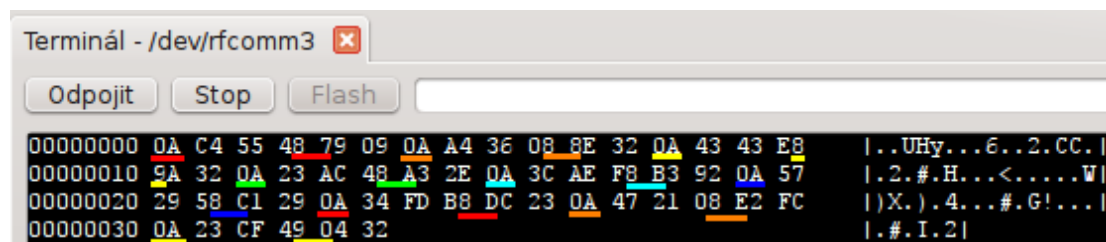


Obrázek 16: Barva v modulu Analyzér

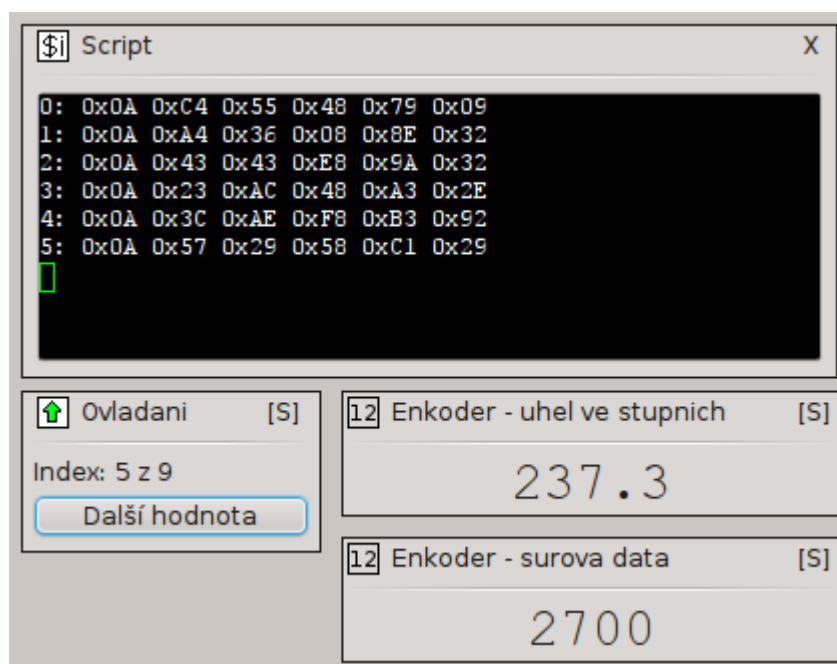
6.2 Testování enkodérů

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 Loris otevřu modul analyzátor a ve widgetu „script“ napíši jednoduchý script, který složí úhel do jednoho čísla a zobrazí ho ve widgetu „číslo“.



Obrázek 17: Surová data z enkodérů s vyznačenými byte, které obsahují úhel natočení



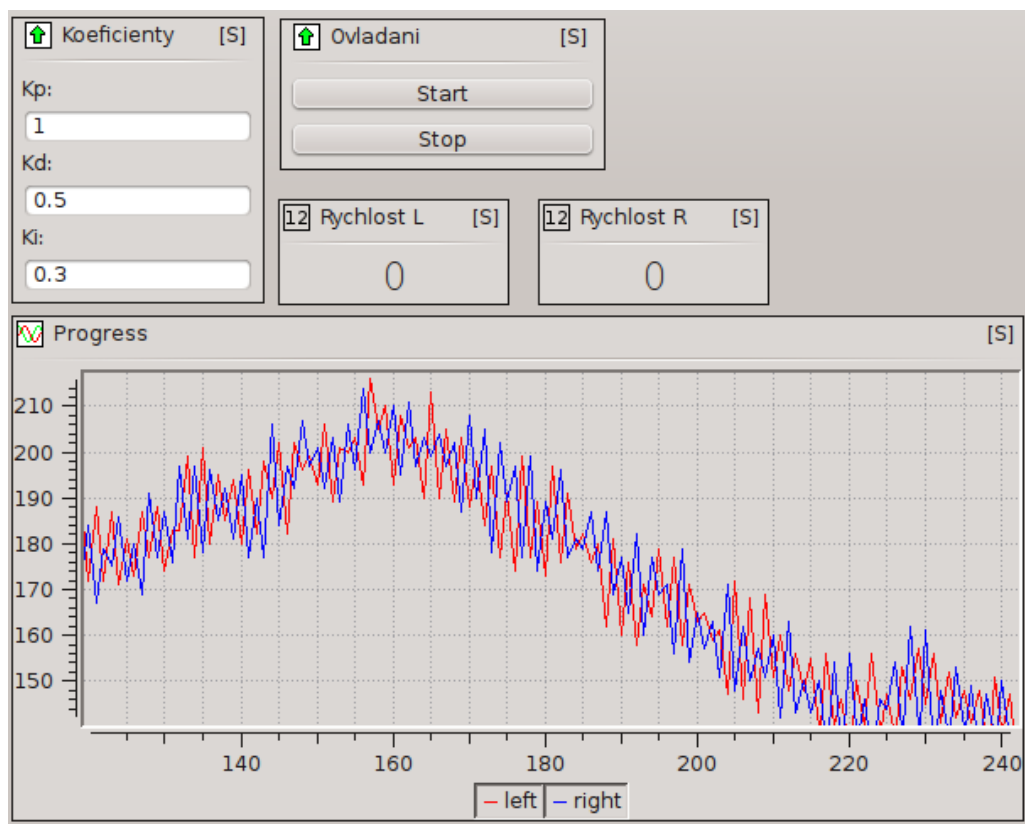
Obrázek 18: Data z enkodérů zpracovaná analyzérem

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 aktuální 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 napíši jednoduchý script, který po stisku kláves změní nastavení konstant regulátoru nebo rozjede/zastaví robota.



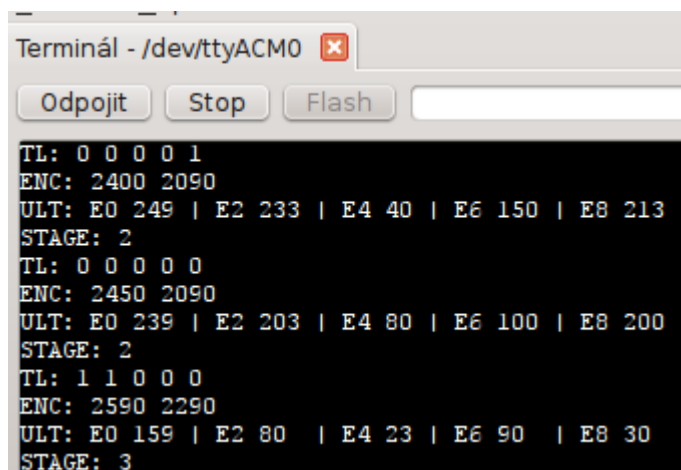
Obrázek 19: Ladění PID regulátoru

6.4 Vývoj robota pro soutěž Eurobot 2011

V minulém roce jsem se zúčastnil soutěže Eurobot[9]. 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). Kompletní pravidla, výsledková listina a další informace jsou na webu ročníku 2011[10].

Robot našeho týmu byl poměrně jednoduchý, přesto však obsahoval 5 ul-

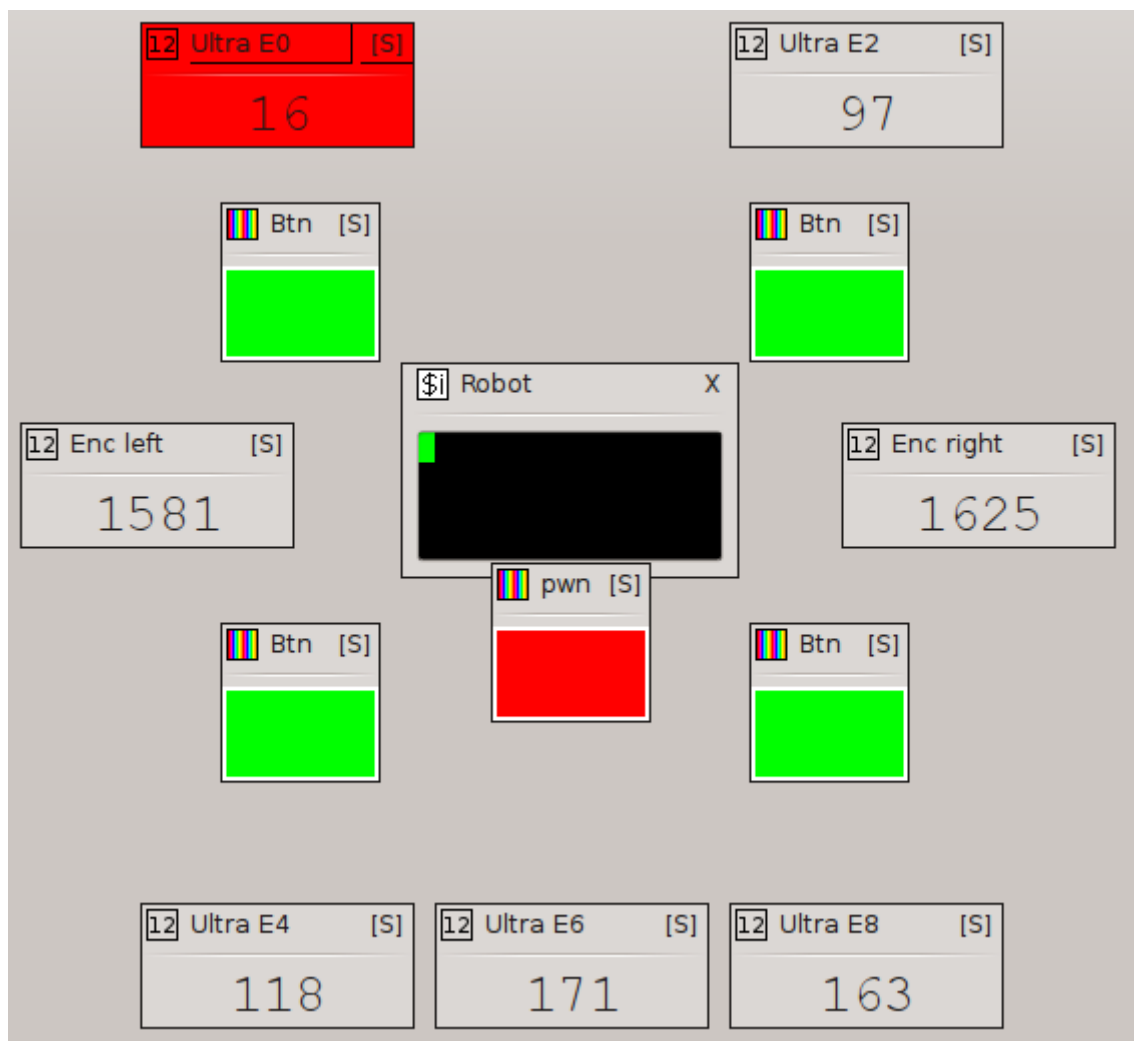
trazvukových měřáků vzdálenosti, dva enkodéry a 5 tlačítek (detekce nárazu na mantinel a pěšce, kterého mohl robot převážet). Tyto senzory produkují poměrně značné množství dat, které se v terminálu zobrazuje nepřehledně.



```
Terminál - /dev/ttyACM0
Odpojit Stop Flash
TL: 0 0 0 0 1
ENC: 2400 2090
ULT: E0 249 | E2 233 | E4 40 | E6 150 | E8 213
STAGE: 2
TL: 0 0 0 0 0
ENC: 2450 2090
ULT: E0 239 | E2 203 | E4 80 | E6 100 | E8 200
STAGE: 2
TL: 1 1 0 0 0
ENC: 2590 2290
ULT: E0 159 | E2 80 | E4 23 | E6 90 | E8 30
STAGE: 3
```

Obrázek 20: 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 by se mohla tato data zobrazovat například takto:



Obrázek 21: Simulovaná data v analyzáru

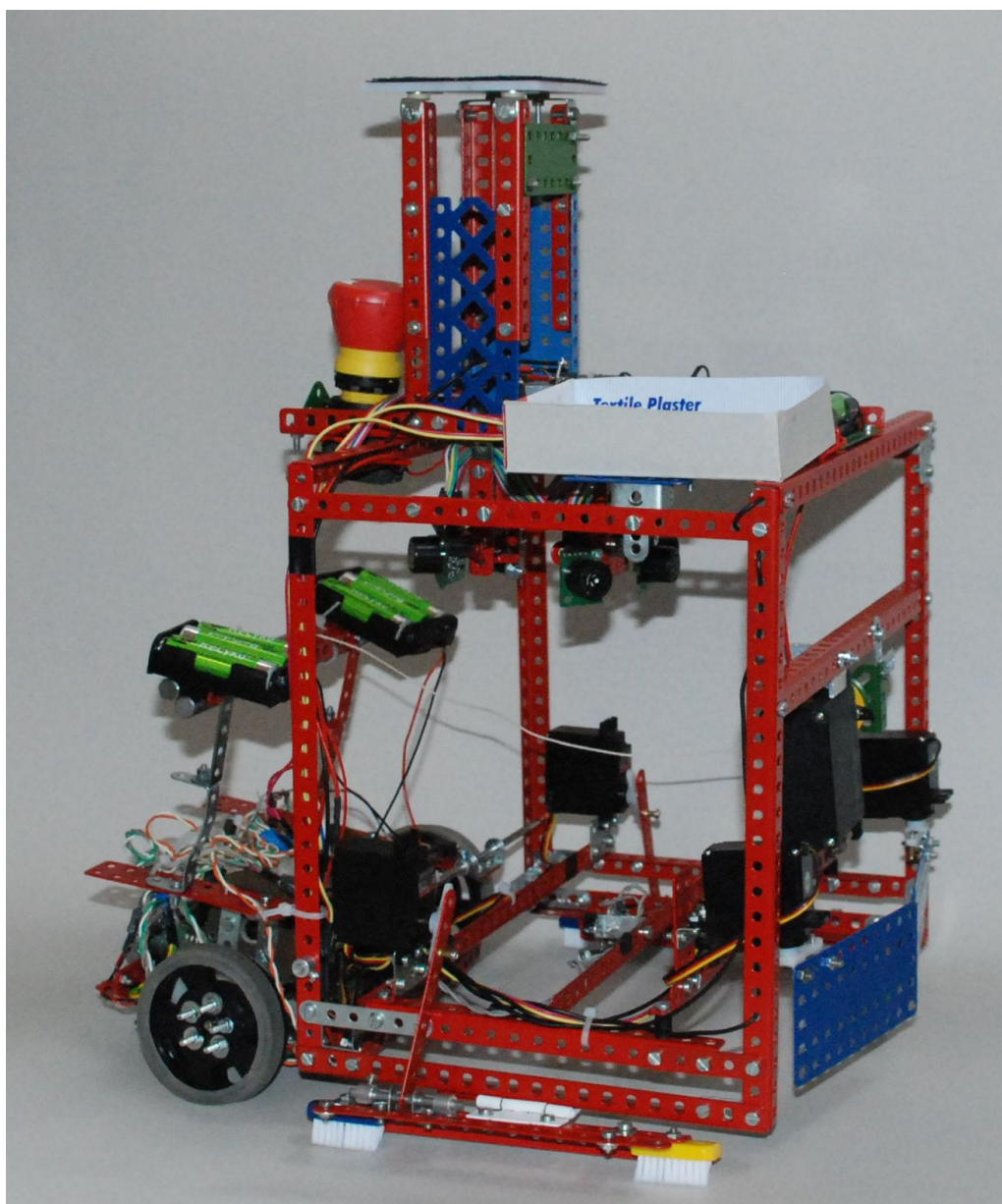
Všechny widgety jsou rozmístěné stejně jako na robotovi – 2 ultrazvuky měří vzdálenost vpředu, 3 vzadu; tlačítka pro detekci mantinelu jsou na každém rohu, tlačítko uvnitř robota ověřuje, zda robot nabral pěšce a enkodéry na obou kolech měří ujetou vzdálenost.

Widget *script* s názvem „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 hra-

niční 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.



Obrázek 22: Náš robot *David* skončil na 4. místě v soutěži Eurobot 2011

7 Knihovny třetích stran, licence

7.1 Knihovny třetích stran

- **Qwt**[11] je knihovna pro Qt Framework obsahující tzv. widgety pro aplikace technického charakteru – grafy, sloupcové ukazatele, kompasy a podobně. Ve svojí práci zatím z této knihovny používám pouze graf (v modulu analyzáru).
- **QExtSerialPort**[12] poskytuje připojení k sériovému portu a také dokáže vypsat seznam nalezených portů v počítači.
- **QHexEdit2**[13] je hex editor použitý v modulu programátoru Shupito na zobrazování obsahu paměti. V této knihovně jsem upravoval několik málo drobností, týkajících se především vzhledu.

7.2 Licence

Lorris je dostupný pod licencí GNU GPLv3[14], licence použitých programů a knihoven jsou následující:

- **Qt Framework** je distribuován pod licencí GNU LGPLv2.1[15]
- **Qwt** je distribuováno pod Qwt license[16], která je založená na GNU LGPLv2.1
- **QExtSerialPort** je distribuován pod The New BSD License[17]
- **QHexEdit2** je distribuován pod licencí GNU LGPLv2.1
- **avr232client** je distribuován pod licencí Boost Software License v1.0[18]

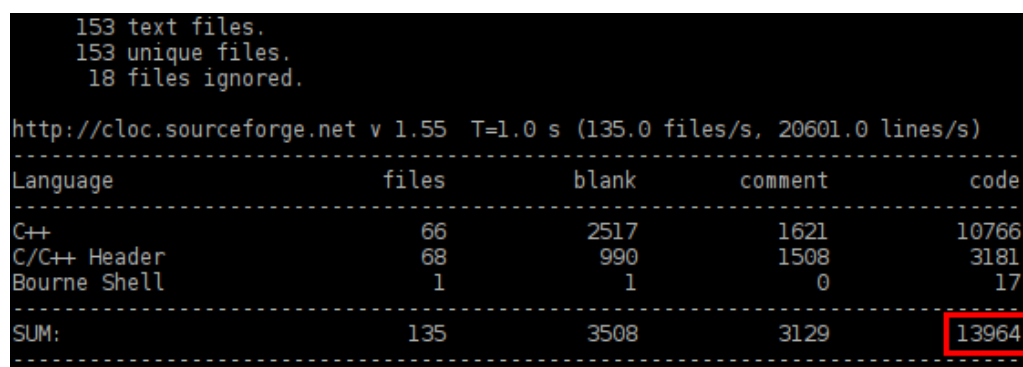
Všechny tyto licence umožňují svobodné používání a šíření kódu.

Závěr

Vytvořená aplikace splňuje všechny stanovené požadavky:

- ✓ 1. Možnost zpracovávat data přicházející ze zařízení a přehledně je zobrazovat
- ✓ 2. Podpora co nejvíce formátů příchozích dat
- ✓ 3. Snadné a rychlé používání
- ✓ 4. Možnost běhu i na jiných systémech než je MS Windows
- ✓ 5. Co možná nejnížší cena
- ✓ 6. Snadná rozšiřitelnost (ideálně otevřený zdrojový kód)
- ✓ 7. Nezávislost na další aplikaci (např. MS Office Excel)

Aplikace je nadále vyvíjena, můžu prakticky donekonečna přidávat buďto další typy widgetů do modulu Analyzátor (například kompas, směrový kříž, ...) nebo celé nové moduly (například ovládání robota pomocí joysticku). Počet řádků kódu v programu je v současné době (8.3.2012) tento:



```
153 text files.
153 unique files.
18 files ignored.

http://cloc.sourceforge.net v 1.55 T=1.0 s (135.0 files/s, 20601.0 lines/s)
-----
Language             files      blank      comment      code
-----
C++                   66         2517         1621        10766
C/C++ Header          68          990         1508         3181
Bourne Shell           1            1            0            17
-----
SUM:                  135         3508         3129        13964
-----
```

Obrázek 23: Počet řádků spočítaný programem CLOC[19] (bez knihoven třetích stran)

Kromě přidávání dalších vlastností do tohoto programu bych v budoucnu rád vytvořil podobný program (hlavně vlastnosti modulu Analyzér) pro přenosná zařízení (chytrý mobilní telefon či tablet), protože pro tyto zařízení žádná taková aplikace v současné době neexistuje a chtěl bych vyzkoušet programování pro tyto platformy (zejména pro Google Android[20]).

PŘÍLOHA A: Podrobný popis widgetu *script* z modulu Analyzér

Script widget umožňuje parsování dat pomocí scriptu, který se píše v Qt-Scriptu, který je založený na standartu ECMAScript, na kterém je založený JavaScript. Prostě je to hodně podobné JavaScriptu a většinou můžete použít jeho referenci. Tento text předpokládá alespoň základní znalost JavaScriptu nebo podobného programovacího jazyku.

- <http://en.wikipedia.org/wiki/ECMAScript>
- <https://qt-project.org/doc/qt-4.8/scripting.html>
- <http://www.w3schools.com/jsref/default.asp> - JS reference

Pár věcí na které je třeba myslet

- Widgets vytvořené ze scriptu se neukládají do datového souboru - po načtení se vytvoří znovu, bez dat.
- Stav proměnných ve scriptu se zatím (?) neukládá do souboru.
- Po stisknutí "Ok" nebo "Použít" v dialogu nastavení scriptu se script načte znovu - staré widgety se smažou a vytvoří nové, bez dat.
- Jazyk nemá žádné pojistky proti „špatnému“ kódu – pokud ve scriptu bude nekonečná smyčka, Loris prostě zamrzne

Základní script

Script by měl obsahovat následující dvě funkce (ale nemusí, pokud je nepoužívá):

```
1 function onDataChanged(data, dev, cmd, index) {  
2     return "";  
3 }  
4  
5 function onKeyPress(key) {  
6  
7 }
```

Příklad 1: Základní script

`onDataChanged(data, dev, cmd, index)` je volána při změně pozice v datech (tj. když přijdou nová data nebo uživatel pohne posuvníkem historie). Může vrátit **string**, který se přidá do terminálu.

- **data** – **pole s Integery** obsahující příchozí data
- **dev** – **Integer** s ID zařízení (může být definováno v hlavičce packetu – pokud není, **dev** se rovná -1)
- **cmd** – **Integer** s ID příkazu (může být definováno v hlavičce packetu – pokud není, **cmd** se rovná -1)
- **index** – **Integer** s indexem packetu v příchozích datech.

`onKeyPress(key)` je volána po stisku klávesy v terminálu. Tato funkce by neměla vrátit nic.

- **key** – **String** se stisknutou klávesou

Dostupné funkce

Jsou dostupné základní javascriptové knihovny (**Math**, **Date**, ...), jejich reference najdete na internetu. Samotný **Lorris** poskytuje další rozšiřující funkce.

- `appendTerm(string)` – přidá do terminálu text.

```
1 function onKeyPress(key) {
2     appendTerm(key); // vypise _key_ do terminalu
3 }
```

Příklad 2: Vypsání stisknutých kláves do terminálu

- `clearTerm()` – vyčistí terminál.

```
1 function onKeyPress(key) {
2     if(key == "c")
3         clearTerm(); // vycisti terminal
4     else
5         appendTerm(key); // vypise _key_ do terminalu
6 }
```

Příklad 3: Vypsání stisknutých kláves do terminálu a jeho vyčištění po stisku klávesy C

- `sendData(pole Integerů)` – pošle data do zařízení

```
1 function onKeyPress(key) {
2     sendData(new Array(key.charCodeAt(0)));
3 }
```

Příklad 4: Poslání ASCII kódu stisknuté klávesy

- `newXXXXWidget()` – tato funkce potřebuje o něco obsáhlejší popis, který je v následující kapitole

Vytvoření widgetu

Script může vytvořit všechny ostatní typy widgetů a posílat do nich data. Samotných funkcí je několik, každá vytvoří jiný typ widgetu:

- `newNumberWidget(...)` – vytvoří widget *číslo*

- `newBarWidget(...)` – vytvoří widget *sloupcový bar*
- `newColorWidget(...)` – vytvoří widget *barva*
- `newGraphWidget(...)` – vytvoří widget *graf*
- `newInputWidget(...)` – vytvoří widget *vstup* (dostupný pouze ze scriptu)

Všechny tyto funkce mají stejné parametry a vrací object widgetu.

```
newXXXXWidget("jméno");
newXXXXWidget("jméno", šířka, výška);
newXXXXWidget("jméno", šířka, výška, Xoffset, Yoffset);
```

- **jméno** – **String**, jméno widgetu, zobrazí se v titulku
- **šířka** – **Integer**, šířka widgetu v pixelech. Může být 0, poté se zvolí minimální velikost.
- **výška** – **Integer**, výška widgetu v pixelech. Může být 0, poté se zvolí minimální velikost.
- **Xoffset** – **Integer**, vodorovná vzdálenost v pixelech od levého horního rohu mateřského ScriptWidgetu. Může být 0, widget se poté vytvoří v levém horním rohu aktuálně viditelné plochy.
- **Yoffset** – **Integer**, svislá vzdálenost v pixelech od levého horního rohu mateřského ScriptWidgetu. Může být 0, widget se poté vytvoří v levém horním rohu aktuálně viditelné plochy.

```

1 var cislo = newNumberWidget("rychlost", 200, 100, -250, 0);
2
3 function onDataChanged(data, dev, cmd, index) {
4     cislo.setValue(data[0]);
5     return "";
6 }

```

Příklad 5: Vytvoření widgetu *číslo* a nastavení jeho hodnoty z příchozích dat

Dostupné funkce widgetů

Objekt widgetu je podtřídou třídy z Qt Frameworku QWidget - díky tomu může používat jeho vlastnosti a sloty. Popis vlastností najdete v Qt referenci¹³ v kapitole „Properties“ a ve scriptu se používají takto:

```

1 var cislo = newNumberWidget("rychlost", 200, 100, -250, 0);
2 cislo.width = 40; // nastaveni sirky widgetu

```

Příklad 6: Vytvoření widgetu *číslo* a nastavení vlastnosti „width“

Popis slotů je také v Qt referenci, tentokrát pod kapitolou „Public slots“. Používají se jako metody:

```

1 var cislo = newNumberWidget("rychlost", 200, 100, -250, 0);
2 cislo.setDisabled(true); // znemozneni interakce s widgetem

```

Příklad 7: Vytvoření widgetu *číslo* a použití slotu

Kromě těchto zděděných vlastností a funkcí má každý typ widgetu své vlastní.

¹³<http://qt-project.org/doc/qt-4.7/qwidget.html#propertySection>

Widget číslo

- `setValue(Integer nebo double)` - Nastaví hodnotu widgetu

```
1 var cislo = newNumberWidget("test cislo", 200, 100, -250, 0);
2 cislo.setValue(40);
3 ...
4 cislo.setValue(3.14);
```

Příklad 8: Nastavení hodnoty widgetu *číslo*

Widget sloupcový bar

- `setValue(Integer)` - Nastaví hodnotu widgetu
- `setRange(Integer min, Integer max)` - Nastaví minimální a maximální hodnotu widgetu
- `rotationSelected(Integer)` - Nastaví rotaci sloupce. 0 pro svislou, 1 pro vodorovnou

```
1 var bar = newBarWidget("test bar");
2 bar.setRange(0, 100); // rozmezi hodnot 0 az 100
3 bar.setValue(45); // nastaveni hodnoty na 45
4 bar.rotationSelected(1); // otoceni na vodorovno
```

Příklad 9: Nastavení hodnot widgetu *sloupcový bar*

Widget barva

- `setValue(Integer r, Integer g, Integer b)` - Nastaví barvu z hodnot RGB (každá 0 až 255)
- `setValue(String barva)` - Nastaví barvu z hex hodnoty jako v HTML (např. #FF0000 pro červenou)


```

1 var bar = newColorWidget("test barva");
2 bar.setValue(255, 255, 0);
3 ...
4 bar.setValue("#00FF00");

```

Příklad 10: Nastavení hodnot widgetu *barva*

Widget graf

Tento widget se od ostatních poměrně výrazně liší - je třeba nejdříve vytvořit křivku až te nastavovat hodnoty. Funkce samotného widgetu graf jsou tyto:

- `addCurve(String jméno, String barva)` – Vytvoří a vrátí novou křivku. `barva` může být buďto html název (např. red, blue) nebo HTML hex zápis (např. #FF0000)
- `setAxisScale(bool proX, double min, double max)` – Nastaví měřítko os. `proX` je **true** pokud nastavujete měřítko osy *x*
- `updateVisibleArea()` – Přesune pohled na nejvyšší hodnotu osy *x*

`addCurve(String jméno, String barva)` vrátí křivku, která má tyto funkce:

- `addPoint(Integer index, double hodnota)` – Vloží bod křivky. `index` určuje pořadí bodů (bod s indexem 0 bude vždy před bodem s indexem 50, i když bude vložen až po něm). Pokud bod se stejným indexem už existuje, je jeho hodnota změněna
- `clear()` – Smaže všechny body křivky

```

1 var graf = newGraphWidget("graf", 400, 250, -420, 0);
2 graf.setAxisScale(false, -105, 105); // merito osy y
3 graf.setAxisScale(true, 0, 200); // merito osy x
4
5 // vytvoreni krivky sin
6 var sin = graf.addCurve("sin", "blue");
7
8 // pridani bodu do krivky sin
9 var sinVal = 0;
10 for(var i = 0; i < 500; ++i)
11 {
12     sin.addPoint(i, Math.sin(sinVal)*100);
13     sinVal += 0.1;
14 }
15 // presunuti na posledni hodnotu krivky
16 graf.updateVisibleArea();

```

Příklad 11: Zobrazení křivky funkce sinus ve widgetu *graf*

Widget vstup

Tento widget lze vytvořit pouze ze scriptu a umí zobrazit a ovládat většinu Qt widgetů¹⁴, například tlačítko (QPushButton), zaškrtačací políčko (QCheckBox) či textové políčko (QLineEdit). Dokumentace k těmto widgetům je v Qt referenci, opět můžete používat vlastnosti („Properties“) a funkce („Public slots“).

Funkce widgetu *vstup*:

- `newWidget(String jméno, Integer roztahování = 0)` – Vytvoří a vrátí nový QWidget. *jméno* musí být jméno třídy widgetu, například QPushButton, QCheckBox nebo QLineEdit. *roztahování* značí jak moc se bude widget roztahovat oproti ostatním.

¹⁴<http://qt-project.org/doc/qt-4.7/widgets-and-layouts.html>

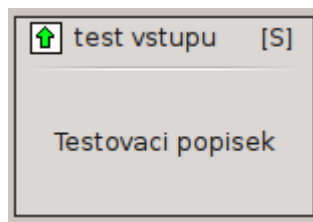
```

1  var vstup = newInputWidget("test vstupu", 150, 100, -160, 0);
2  var label = vstup.newWidget("QLabel", 1);
3
4  // zarovnani textu na stred.
5  // 0x0080 a 0x0004 jsou konstanty Qt Frameworku
6  // Qt::AlignHCenter a Qt::AlignVCenter
7  label.alignment = 0x0080 | 0x0004;
8
9  // nastaveni textu
10 label.text = "Testovací popisek";

```

Příklad 12: Widget *vstup* – vytvoření QLabel

Widget vytvořený tímto příkladem vypadá takto:

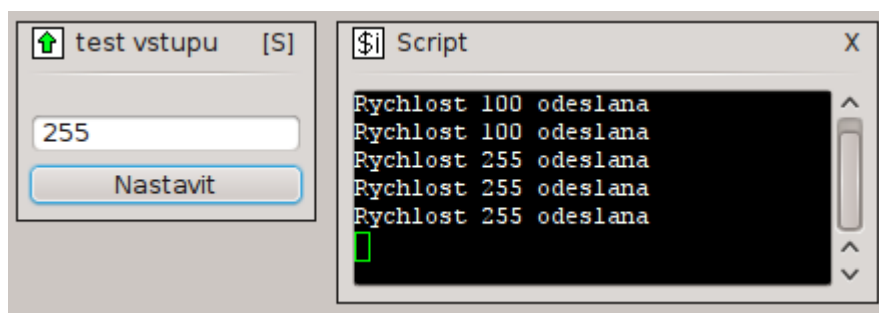


Obrázek 24: Widget *vstup* – vytvoření QLabel

QtScript podporuje i využití principu signálů a slotů, díky tomu lze ve scriptu reagovat například na stisknutí tlačítka.

```
1 var vstup = newInputWidget("test vstupu", 150, 100, -160, 0);
2
3 var rychlost = vstup.newWidget("QLineEdit");
4 rychlost.text = "100";
5
6 var btn = vstup.newWidget("QPushButton", 1);
7 btn.text = "Nastavit";
8
9 function posliRychlost()
10 {
11     var speed = parseInt(rychlost.text);
12     sendData(new Array(speed));
13     appendTerm("Rychlost " + speed + " odeslana\n");
14 }
15 // Pripojeni signalu "clicked" na slot posliRychlost()
16 btn.clicked.connect(posliRychlost);
```

Příklad 13: Widget *vstup* – tlačítko



Obrázek 25: Widget *vstup* – tlačítko

Reference

- [1] *SerialChart* – Analyse and chart serial data from RS-232 COM ports
<http://code.google.com/p/serialchart/>
(Stav ke dni 6.3.2012)
- [2] *WinWedge* – RS232 data collection software
<http://www.taltech.com/products/winwedge/>
(Stav ke dni 6.3.2012)
- [3] *Advanced Serial Data Logger*
<http://www.aggsoft.com/serial-data-logger.htm>
(Stav ke dni 6.3.2012)
- [4] *StampPlot Pro* – Graphical Data Acquisition and Control
<http://www.selmaware.com/stampplot/index.htm>
(Stav ke dni 6.3.2012)
- [5] *Qt* – Cross-platform application and UI framework
<http://qt.nokia.com/>
(Stav ke dni 6.3.2012)
- [6] *Debian Linux* – The Universal Operating System
<http://www.debian.org/>
(Stav ke dni 6.3.2012)
- [7] *GitHub* – Social Coding
<https://github.com>
(Stav ke dni 6.3.2012)
- [8] *avr232client*
<http://technika.junior.cz/trac/wiki/avr232client>
(Stav ke dni 6.3.2012)

- [9] *Eurobot*
<http://www.eurobot.cz/>
(Stav ke dni 6.3.2012)
- [10] *Eurobot 2011*
<http://www.eurobot.cz/eurobot2011.php>
(Stav ke dni 6.3.2012)
- [11] *Qwt – Qt Widgets for Technical Applications*
<http://qwt.sourceforge.net/>
(Stav ke dni 6.3.2012)
- [12] *QExtSerialPort – Qt interface class for old fashioned serial ports*
<http://code.google.com/p/qextserialport/>
(Stav ke dni 6.3.2012)
- [13] *QHexEdit2 – Binary Editor for Qt*
<http://code.google.com/p/qhexedit2/>
(Stav ke dni 6.3.2012)
- [14] *GNU General Public License v3*
<http://gplv3.fsf.org/>
(Stav ke dni 6.3.2012)
- [15] *GNU Lesser General Public License v2.1*
<http://www.gnu.org/licenses/lgpl-2.1.html>
(Stav ke dni 6.3.2012)
- [16] *Qwt license*
<http://qwt.sourceforge.net/qwtlicense.html>
(Stav ke dni 6.3.2012)
- [17] *The New BSD License*
<http://www.opensource.org/licenses/bsd-license.php>
(Stav ke dni 6.3.2012)

- [18] *The Boost Software License*
<http://www.boost.org/users/license.html>
(Stav ke dni 6.3.2012)
- [19] *CLOC* – Count Lines of Code
<http://cloc.sourceforge.net/>
(Stav ke dni 8.3.2012)
- [20] *Google Android* – Operační systém pro chytré telefony
<http://www.android.com/>
(Stav ke dni 8.3.2012)

Seznam obrázků

1	Dialog vytvoření panelu	11
2	Modul analyzér	12
3	Dialog nastavení struktury dat	13
4	Widgety	14
5	Widget: číslo	14
6	Widget: sloupcový bar	15
7	Widget: barva	16
8	Widget: graf	17
9	Dialog pro nastavení parametrů křivky grafu	17
10	Widget: script	18
11	Dialog pro nastavení zdrojového scriptu	19
12	Proxy mezi sériovým portem a TCP socketem	20
13	Modul Shupito	21
14	Možnost Shupito Tunel v dialogu připojení k zařízení	22
15	Modul terminál	22
16	Barva v modulu Analyzér	24
17	Surová data z enkodérů s vyznačenými byty, které obsahují úhel natočení	25
18	Data z enkodérů zpracovaná analyzérem	26
19	Ladění PID regulátoru	27
20	Data z robota v terminálu	28
21	Simulovaná data v analyzáru	29
22	Náš robot <i>David</i> skončil na 4. místě v soutěži Eurobot 2011	31
23	Počet řádků spočítaný programem CLOC[19] (bez knihoven třetích stran)	33
24	Widget <i>vstup</i> – vytvoření QLabel	43
25	Widget <i>vstup</i> – tlačítko	44