

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, 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	7
Popis rozhraní	7
Modul: Analyzér	8
Popis	8
Widget: číslo	10
Widget: sloupcový bar	11
Widget: barva	11
Widget: graf	12
Widget: script	13
Modul: Proxy mezi sériovým portem a TCP socketem	14
Popis	14
Modul: Shupito	15
Popis	15
RS232 tunel	16
Modul: Terminál	17
Popis	17
Příklady použití	18
1. Testování barevného senzoru	18
2. Testování enkodérů	19
3. Ladění PID regulátoru	20
4. Vývoj robota pro soutěž Eurobot 2011	21
Knihovny třetích stran, licence	24
Knihovny třetích stran	24
Licence	24
Reference	25

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.

Popis rozhraní

Svůj program jsem pojmenoval "Lorris", je vytvořený v C++ a využívá Qt Framework(v4.7)[1], což je multiplatformní framework, který mimo jiné umožňuje spustit aplikaci na více systémech – testoval jsem na Debian Linux (Wheezy, 64bit) a Windows 7.

Program je navrhnutý jako modulární aplikace, aby mohl zastřešit několik samostatných částí, které však mají podobnou oblast použití. Základní část 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 Shupito)
- TCP socket[2]
- Načtení dat ze souboru

Je možné mít připojeno více různých modulů na jedno zařízení.

Modul: Analyzér



Obrázek 1: Modul analyzér

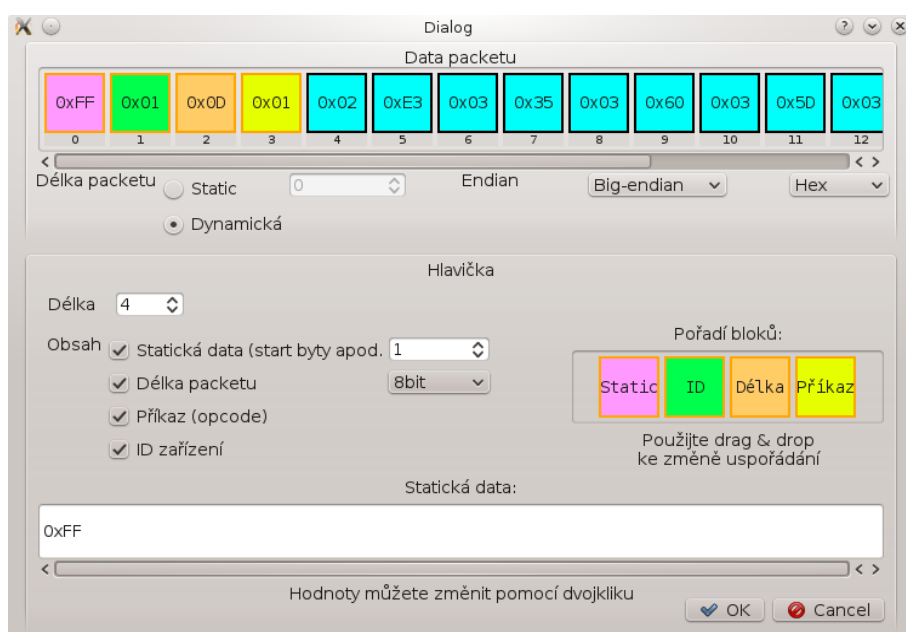
Tento modul parsuje data (strukturované do packetů) přicházející ze zařízení a zobrazuje je v grafických "widgetech". Zpracovaná data si aplikace ukládá do paměti – listování packety je možné pomocí posuvníku a boxu v horní části okna. Data (přijatá data, struktura packetů a rozestavení a nastavení widgetů)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

Struktura dat se nastavuje v samostatném dialogu (viz obrázek 3), kde je možno nastavit délku packetu, jeho endianness[9], 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.

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.

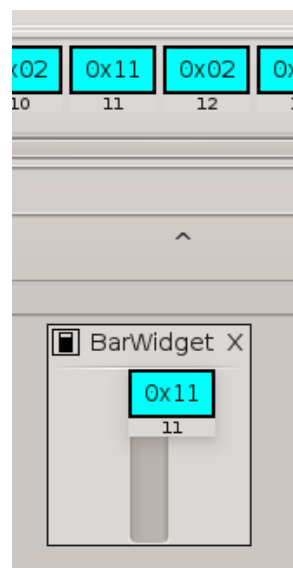
Nastavení widgetu jsou přístupná v kontextovém menu po pravém kliknutí myši na widget. Nastavit lze jméno a další parametry podle typu widgetu – podrobněji jsou možnosti nastavení popsány u jednotlivých widgetů. Widgety je taktéž možné „uzamknout“, aby nebylo možné je zavřít, měnit jejich pozici a velikost.



Obrázek 2: Dialog nastavení struktury dat



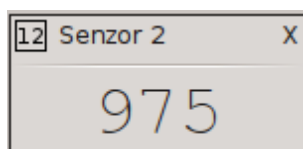
(a) Seznam widgetů



(b) Přiřazení dat pomocí drag&drop

Obrázek 3: Widgety

Widget: číslo



Dokáže zobrazovat celá čísla (se znaménkem i bez, 8 až 64 bitů dlouhé) a desetinná čísla (single-precision¹, 32bit a 64bit).

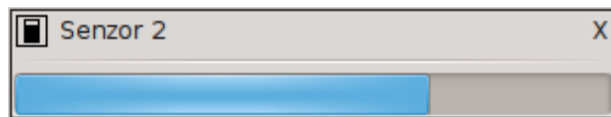
Widget dokáže zarovnat číslo na maximální délku jeho datového typu a formátovat ho těmito způsoby:

- Desítkový – číslo v desítkové soustavě

¹Standartní formát uložení desetinných čísel v jazyku C a dalších (standart IEEE 754-2008).

- 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.

Widget: sloupcový bar



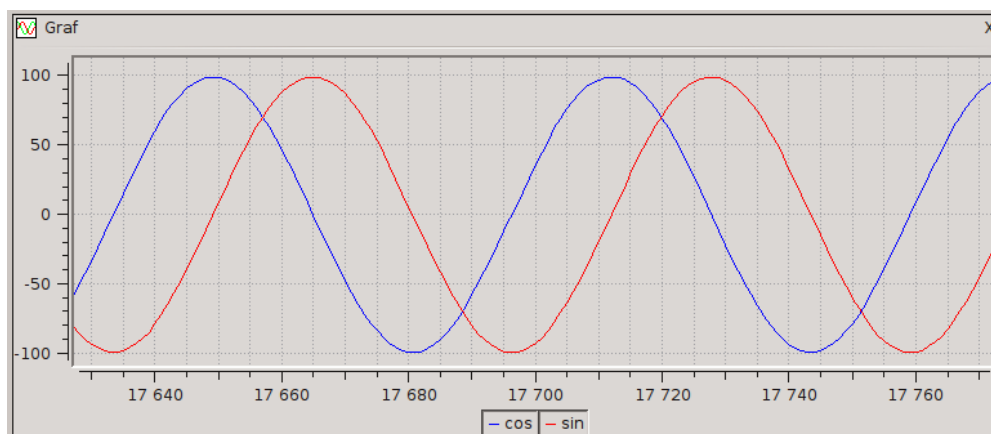
Zobrazuje hodnotu ve sloupcovém baru. Lze nastavit datový typ vstupních dat (stejně jako u čísla), orientaci (vertikální nebo horizontální) a rozmezí zobrazovaných hodnot.

Widget: barva

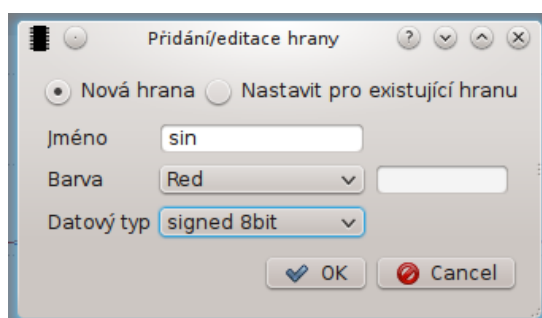


Ukáže 24-bitové hodnoty RGB jako barevný obdélník. Dokáže provést korekci jasu všech barev nebo každé z barev RGB zvlášť.

Widget: graf

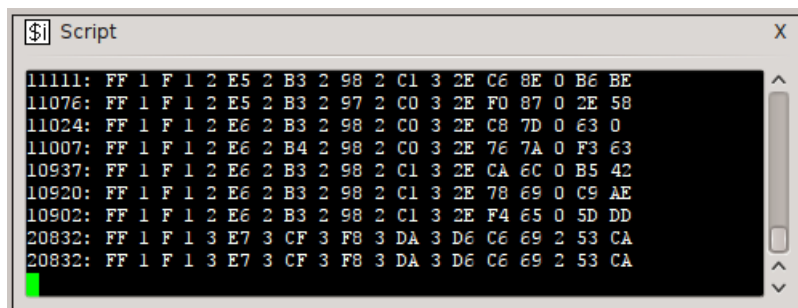


Widget graf zobrazuje hodnoty v grafu – na osu x se vynáší pořadí dat a na osu y hodnoty dat. Lze nastavovat jméno, barvu a datový typ křivky grafu, automatické posouvání grafu, velikost vzorku, měřítko os grafu a zobrazení legendy. Kliknutí na křivku grafu v legendě tuto křivku skryje. Měřítko osy se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

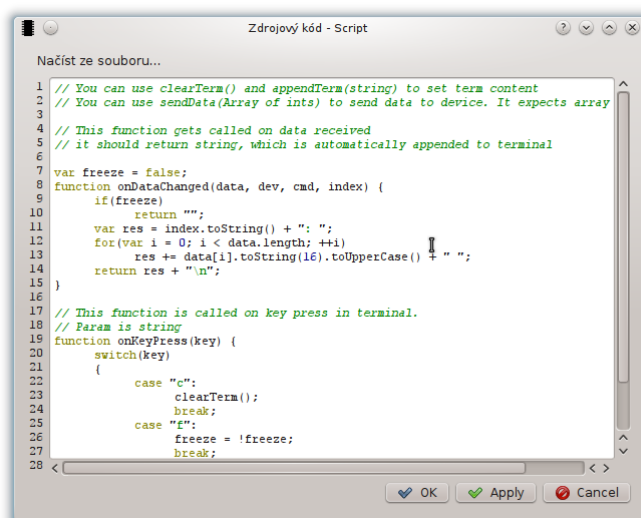


Obrázek 4: Dialog pro nastavení parametrů křivky grafu

Widget: script

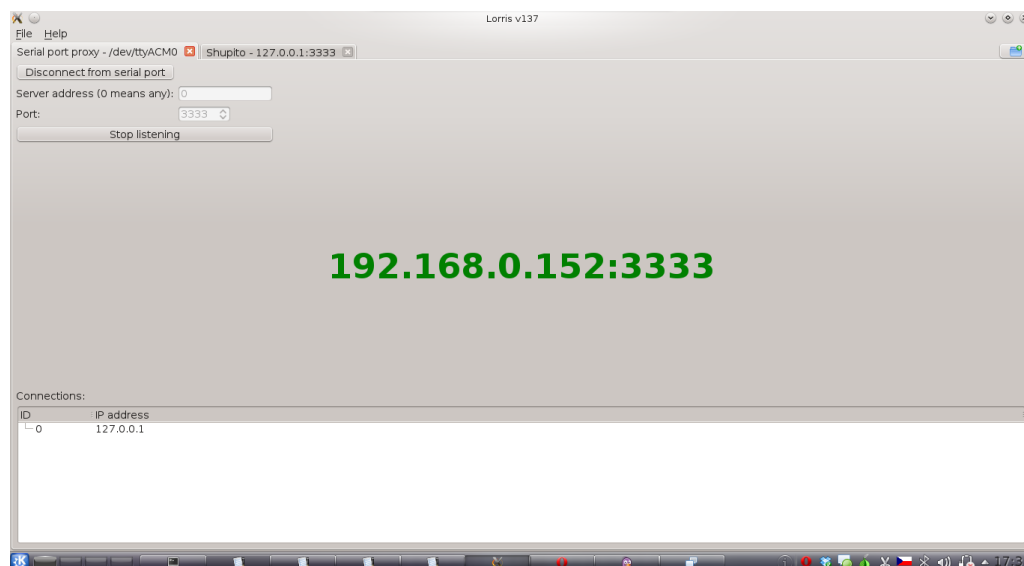


Tento widget umožňuje zpracovávání dat pomocí scriptu, který si napíše sám uživatel. Jazyk, ve kterém se tento script píše je QtScript[10] (jazyk založený na standartu ECMAScript[11], stejně jako JavaScript[12], díky tomu jsou tyto jazyky velmi podobné). Script může zpracovávat příchozí data, reagovat 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, ...).



Obrázek 5: Dialog pro nastavení zdrojového scriptu

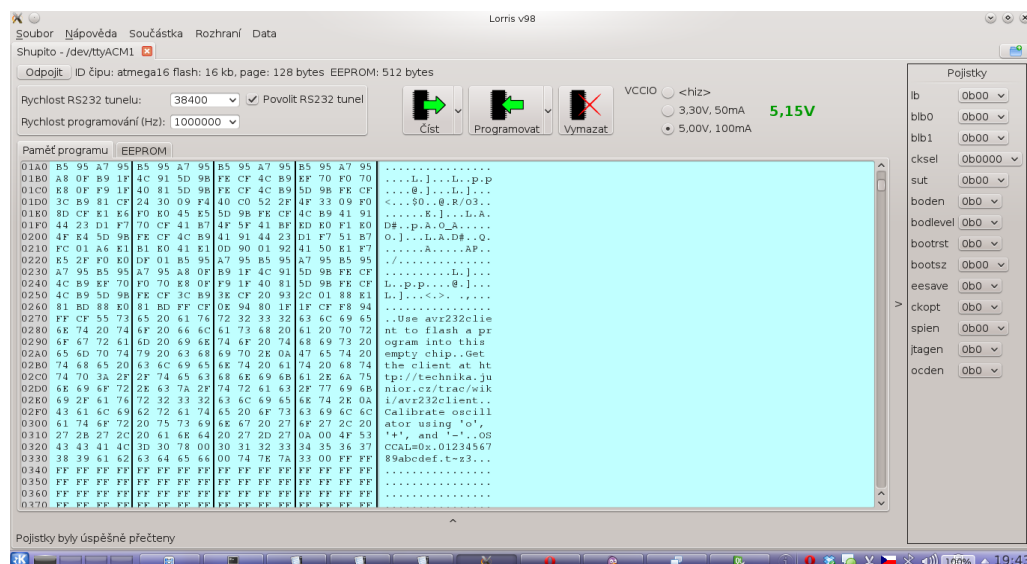
Modul: Proxy mezi sériovým portem a TCP socketem



Obrázek 6: Proxy mezi sériovým portem a TCP socketem

Jednoduchá proxy mezi sériovým portem a TCP socketem. Vytvoří jednoduchý 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.

Modul: Shupito



Obrázek 7: Modul Shupito

Shupito je programátor mikročipů vytvořený Martinem Vejnárem, který dokáže programovat mikrokontroléry pomocí ISP², PDI³ a JTAG⁴ 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[14]. Způsob komunikace s programátorem je přenesen z oficiálního ovládacího programu[8], který je však na rozdíl od Lorris dostupný pouze pro MS Windows.

²*In-system programming* – rozhraní, které umožňuje programovat čipy bez dalšího zařízení přímo v desce plošného spoje.

³*Program and Debug Interface* – rohraní firmy Atmel umožňující programování čipů přímo na desce, podobně jako ISP

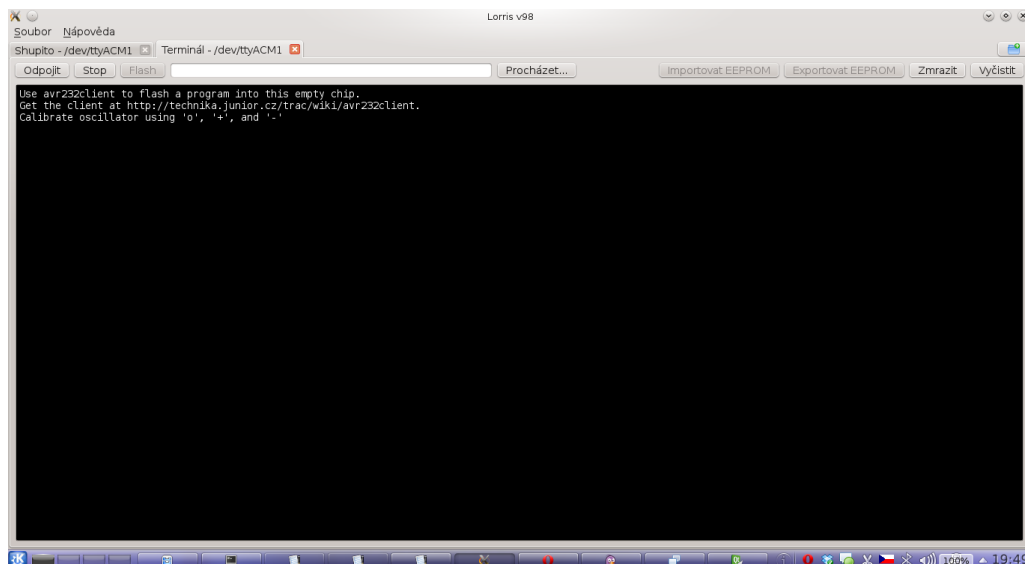
⁴*Joint Test Action Group* – rozhraní podle standartu IEEE 1149.1 umožňující mimo jiné programování a debugování čipů

RS232 tunel

Shupito dokáže vytvořit tunel⁵ 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.

⁵Přímé spojení programovaného čipu a počítače přes programátor.

Modul: Terminál



Obrázek 8: Modul terminál

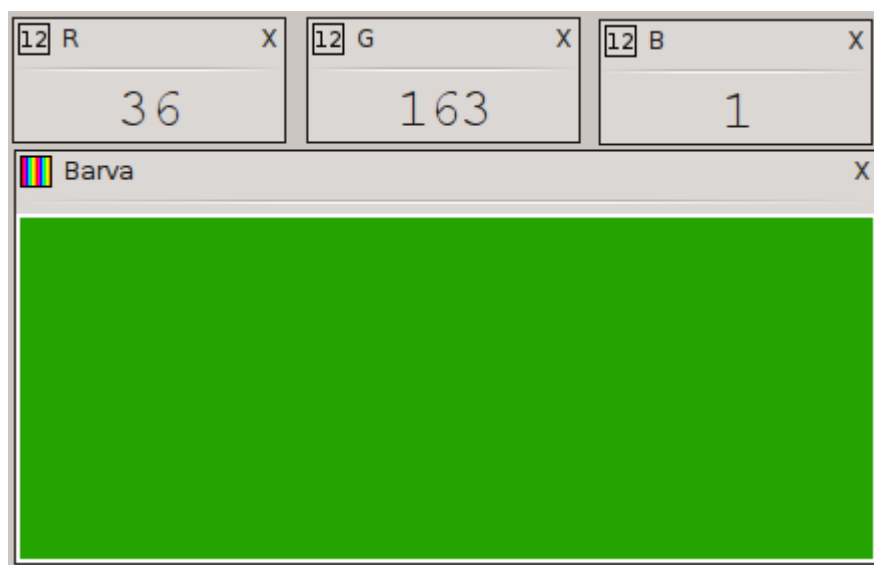
Klasický terminál – zobrazuje data přijatá přes sériový port a posílá stisky kláves. Je obohacený o podporu bootloaderu pro mikrokontroléry AVR ATmega(bootloader byl taktéž napsaný Martinem Vejnardem), který umožňuje jejich programování přes RS232 linku. Informace o protokolu bootloaderu jsem získal z oficiálního programu určeného k programování přes tento bootloader, avr232client[8].

Příklady použití

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 9: Barva v modulu Analyzér

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" napíšu jednoduchý script, který složí úhel do jednoho čísla a zobrazí ho ve widgetu "číslo".

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.

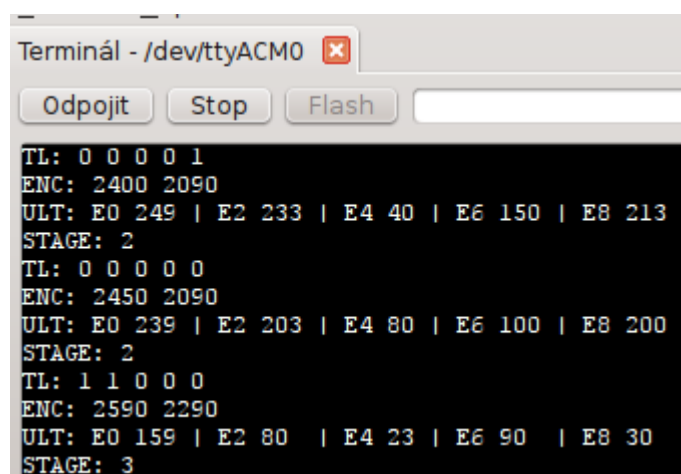
Řešení: Program v robotovi mi posílá aktuální výkon motorů a nastavení konstant PID regulátoru a umožňuje přenastavení těchto konstant a ovládání robota. Tento program do robota nahrávám přes bluetooth pomocí modulu Terminál, protože čip má v sobě bootloader – díky tomu nemusím mít připojený programátor.

V modulu analyzátor 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.

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).

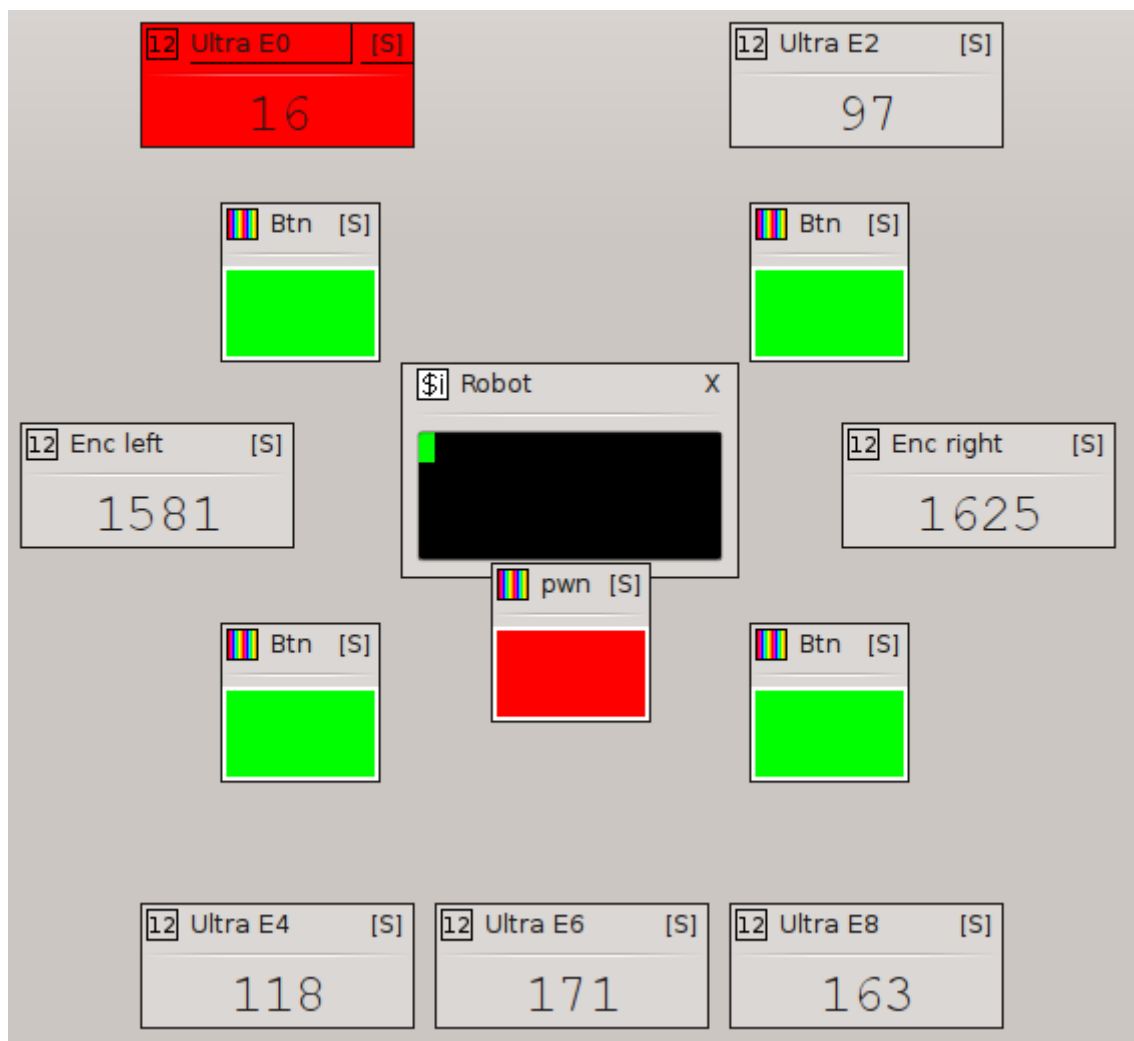
Robot našeho týmu byl poměrně jednoduchý, přesto však obsahoval 5 ultrazvukových měřáků vzdálenosti, dva enkodéry a 5 tlačítek (detekce nárazu na mantinel a pěšce, který mohl robot převážet). Tyto senzory produkují poměrně značné množství dat, které se v terminálu zobrazuje nepřehledně.



```
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 10: Data z robota v terminálu

Toto byl také jeden z důvodů pro výběr toho tématu práce. S použitím programu Lorris vypadá výstup následovně:



Obrázek 11: Data z robota v analyzáru

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 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 "Enc left" a "Enc right" vypisují ujetou vzdálenost z enkodérů pravého a levého kola.

Všechny widgety jsou napozicované stejně jako na robotovi - 2 ultrazvuky měří vzdálenost vepředu, 3 vzadu; tlačítka pro detekci mantinelu jsou na každém rohu atd.

Knihovny třetích stran, licence

Knihovny třetích stran

Qwt[4] je knihovna pro Qt Framework obsahující tzv. widgety pro aplikace technického charakteru – grafy, sloupcové ukazatele, kompas a podobně. Ve svojí práci zatím z této knihovny používám pouze graf (v modulu analyzáru).

QExtSerialPort[5] poskytuje připojení k sériovému portu a také dokáže vypsat seznam nalezených portů v počítači.

QHexEdit2[6] je hex editor použitý v modulu programátoru Shupito na zobrazování obsahu paměti. V této knihovně jsem upravoval několik málo drobností, týkajících se především vzhledu.

Licence

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

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

Reference

- [1] *Qt Framework – Cross-platform application and UI framework*
<http://qt.nokia.com/> (Stav ke dni 28.1.2012)
- [2] *TCP socket*
http://en.wikipedia.org/wiki/Transmission_Control_Protocol
(Stav ke dni 25.2.2012)
- [3] *GIT repozitář Lorris*
<https://github.com/Tassadar/Lorris>
- [4] *Qt Widgets for Technical Applications*
<http://qwt.sourceforge.net/> (Stav ke dni 22.2.2012)
- [5] *Qt interface class for old fashioned serial ports*
<http://code.google.com/p/qextserialport/>
(Stav ke dni 22.2.2012)
- [6] *Binary Editor for Qt*
<http://code.google.com/p/qhexedit2/> (Stav ke dni 22.2.2012)
- [7] *Shupito – Programátor*
<http://shupito.net/> (Stav ke dni 28.1.2012)
- [8] *avr232client*
<http://technika.junior.cz/trac/wiki/avr232client>
(Stav ke dni 28.1.2012)
- [9] *Endianness*
<http://en.wikipedia.org/wiki/Endianness> (Stav ke dni 28.1.2012)
- [10] *QtScript – Making Applications Scriptable*
<http://developer.qt.nokia.com/doc/qt-4.8/scripting.html>
(Stav ke dni 26.2.2012)

- [11] *ECMAScript*
<http://en.wikipedia.org/wiki/ECMAScript> (Stav ke dni 26.2.2012)
- [12] *JavaScript*
<http://en.wikipedia.org/wiki/JavaScript> (Stav ke dni 26.2.2012)
- [13] *Single-precision floating-point format*
http://en.wikipedia.org/wiki/Single_precision (Stav ke dni 28.1.2012)
- [14] *Intel HEX*
http://en.wikipedia.org/wiki/Intel_hex (Stav ke dni 28.1.2012)