Návrh univerzální aplikace pro řízení robotů

Jan Pokorný
Katedra aplikované elektroniky a telekomunikací
Fakulta elektrotechnická
Západočeská univerzita v Plzni
jpokorn@kae.zcu.cz

Design of Universal Application for Robot Control

Abstract – **This article describes a design of a program for easy robot control and for visualization of available sensors. The ConVis Studio, as the program was called, was designed as a universal application, where a user can choose exactly what he wants to visualize and control. It was also necessary to create a new communication protocol for the purposes of this work. The protocol is optimized for the CAN Bus and therefore the application is able to control and visualize not only robots, but everything what implements this protocol within an appropriate range.**

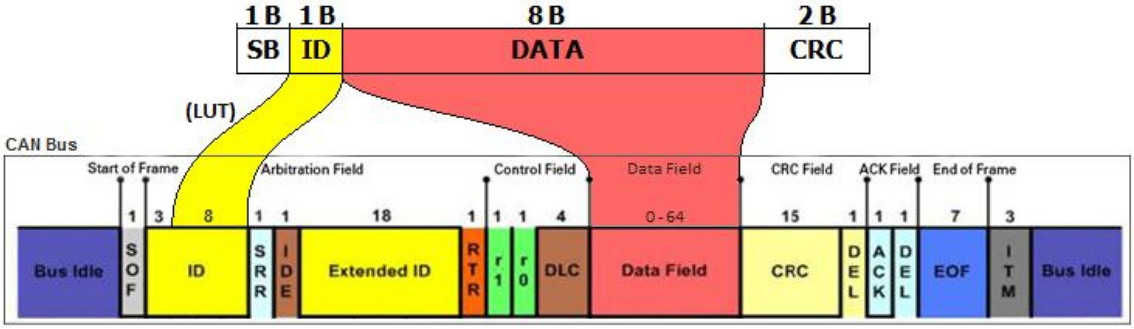
Keywords – application; CAN, communication; control; ConVis Studio; C#; Microsoft Visual Studio; program; robot; vehicles; visualization; .NET Framework

I. ÚVOD

Na Katedře aplikované elektroniky a telekomunikací (KAE) je k dispozici několik robotických platforem: pásová platforma (Tank), terénní čtyřkolka (Crawler), vzducholoď, model kolejiště a další rozpracované platformy. Pro všechny tyto platformy bylo potřeba sjednotit komunikační protokol a vytvořit vhodný řídicí program. Hlavní kroky vzniku celé aplikace budou postupně popsány v tomto článku. Podrobný popis vývoje, včetně uživatelského průvodce prostředím aplikace, je možno nalézt v seznamu literatury pod číslem [1].

II. KOMUNIKAČNÍ PROTOKOL

Komunikační protokol byl optimalizován pro sběrnici CAN, která umožňuje přenést až 8 bajtů dat v jednom rámci. Vlastní protokol tedy fixně definuje 8 bajtů dat a redukuje velikost identifikátoru z 11 (případně 29) bitů na 1 bajt. ID ovšem nemusí být přeložen na CAN ID přímo, ale prostřednictvím překladové tabulky. Obrázek níže ukazuje celý formát protokolu i jeho transformaci na protokol CAN.



Obrázek I. **Transformace komunikačního protokolu na CAN protokol**

Protokol začíná vždy polem „Start-Byte“ (SB), úvodní bajtovou sekvencí střídající dvě jedničky a dvě nuly (0xCC). Následuje jednobajtový identifikátor, který určuje, o jaká data se jedná. Dalšími v pořadí jsou samotná data, která jsou přenášena jako little-endian. Nakonec je paket zakončen kontrolním 16 bitovým cyklickým součtem (CRC). Identifikátor (pole ID) dělí data do 256 možných kategorií takto:

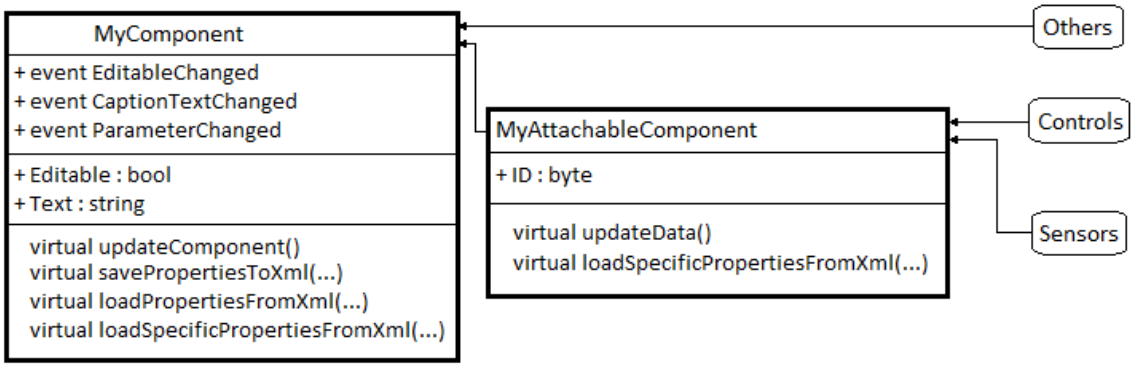
- ID 0 (0x00) je zakázáno využívat,
- ID 1 (0x01) až 19 (0x13) slouží pro zprávy ze zařízení do řídicí aplikace,
- ID 20 (0x14) až 39 (0x27) slouží pro zprávy z řídicí aplikace do zařízení,
- ID 40 (0x28) až 69 (0x45) je vyhrazeno pro řízení akčních členů,
- ID 70 (0x46) až 199 (0xC7) je určeno pro data senzorů,
- ID 200 (0xC8) až 255 (0xFF) jsou volně přístupné. Uživatel si zde může navolit vlastní datové struktury.

III. POPIS NÁVRHU APLIKACE

Program byl vytvořen v jazyce C# s použitím .NET Frameworku 4.5 a Microsoft Visual Studia Professional 2012, které poskytlo patřičnou robustnost návrhu i prostředky pro snadné hledání a ladění chyb. Program se skládá ze dvou projektů – z knihovny grafických komponent a samotného návrhu aplikace.

A. Projekt VisualComponents

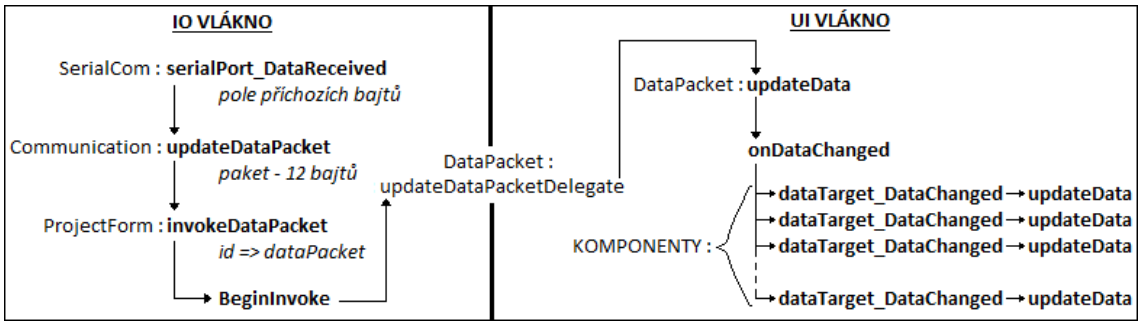
Projekt VisualComponents vytváří DLL knihovnu, která obsahuje grafické komponenty, struktury datových paketů a další potřebné nástroje k popisu a výběru dat v komunikačním protokolu. Vizuální komponenty jsou jednotlivé grafické objekty, simulující senzorické či kontrolní vlastnosti nebo uživatelské prvky. Tyto komponenty jsou rozděleny na ty, které se nemohou napojit na komunikaci, a na ty, které se napojit mohou. Strukturu dědění ukazuje následující obrázek, na kterém jsou vidět i některé důležité události, metody a vlastnosti tříd.



Obrázek II. **Struktura dědění komponent**

B. Projekt ConVis

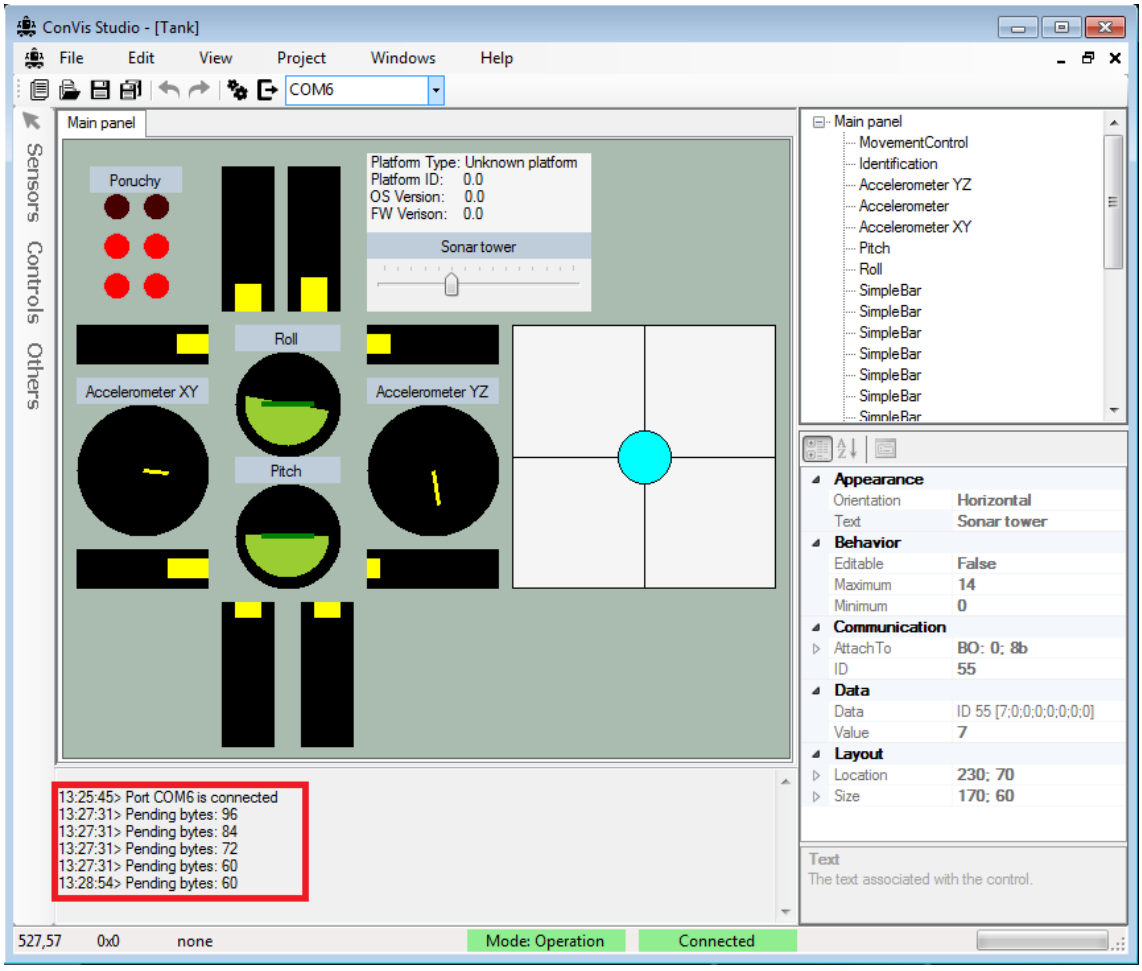
Projekt ConVis, jenž je druhou částí aplikace, tvoří graficko-uživatelské prostředí. Obsahuje hlavní formulář MainMdiForm, formulář projektu ProjectForm, formulář pro nastavení projektu SettingForm a několik dalších méně významných formulářů. Kromě nich má projekt i několik tříd pro správu vizuálních částí studia (konzole, stromový prohlížeč komponent, prohlížeč vlastností komponent a stavový řádek) a třídy pro vytváření projektů studia a třídy pro správu komunikace. Následující obrázek ukazuje proces zpracování paketu z příchozí komunikace až po aktualizaci komponenty.



Obrázek III. **Proces zpracování paketu ze sériové linky**

C. Testování aplikace

ConVis Studio bylo navrženo jako univerzální program pro řízení a vizualizaci s přehledným a jednoduchým graficko-uživatelským rozhraním. Samotná aplikace byla mimo jiné testována i s využitím pásové platformy Tank. Na obrázku níže je ukázka vizualizačního projektu této platformy. Je zde možno nalézt např. osmici dálkových senzorů, gyroskop a akcelerometr, či joystick pro ovládání pohybu robota.



Obrázek IV. **Testování spojení s platformou Tank**

Samotné testy ale přinesly i nepříjemné problémy. Program nebyl schopen všechny příchozí data z Tanku zpracovávat včas. V konzoli v červeném rámečku na předchozím obrázku jsou vidět zprávy z komunikace, informující o nadměrném počtu bajtů čekajících na zpracování. Narůstající počet bajtů po chvíli způsobil „sekání“ programu, a program tak přestal být použitelný. První vylepšení bylo provedeno úpravou datového

toku mezi vlákny (viz obrázek III). Předtím se totiž přepnutí vlákna provádělo až na úrovni komponenty, tedy v metodě updateData. Jak je vidět na obrázku III, metoda byla volána tolikrát, kolik bylo napojeno komponent na stejná data. To vedlo k velkému počtu přepnutí vláken, což je časově náročná operace. Řešením bylo přesunutí přepnutí vlákna až na úroveň projektu.

K další optimalizaci se využil nástroj zvaný „profiling“. Tato funkce spouští program normálním způsobem, ale zároveň monitoruje vytížení procesoru. Výstupem je tedy zpráva, která oznamuje, kolik času procesor strávil nad provedením jednotlivých metod. Využitím této funkce se nalezl výrazný problém s metodou pro obnovou prohlížeče vlastností komponent – prvku PropertyGrid. Téměř 90% času trávil procesor vykonáváním této metody. Srovnání výsledků před vypnutím a po vypnutí metody ukazuje obrázek níže. Pro nynější řídicí aplikace se ukázalo, že předešlé optimalizace jsou již dostatečné.

A: Functions Doing Most Individual Work

Name	Exclusive Samples %
System.Windows.Forms.PropertyGrid.Refresh()	89,13
System.Drawing.Graphics.DrawImage(class System.Drawing.Image,int32,int32,int32,int32)	3,72
System.Windows.Forms.Control.Refresh()	2,97
System.Windows.Forms.Application.Run(class System.Windows.Forms.Form)	1,32
System.Windows.Forms.PropertyGrid.set_SelectedObject(object)	0,71

B: Functions Doing Most Individual Work

Name	Exclusive Samples %
System.Windows.Forms.Control.Refresh()	35,10
System.Drawing.Graphics.DrawImage(class System.Drawing.Image,int32,int32,int32,int32)	29,93
System.Windows.Forms.Application.Run(class System.Windows.Forms.Form)	12,26
System.Drawing.Graphics.DrawLines(class System.Drawing.Pen,valuetype System.Drawing.Point[])	4,81
System.Windows.Forms.PropertyGrid.set_SelectedObject(object)	4,33

Obrázek V. **Srovnání vytíženosti CPU (A: před optimalizací; B: po optimalizaci)**

IV. ZÁVĚR

Cílem práce bylo vytvořit program pro řízení a vizualizaci robotických vozidel. Výstupem této práce je univerzální aplikace ConVis Studio, které je schopno řídit a vizualizovat elektronická zařízení s definovaným komunikačním protokolem. Protože projekt pro grafické komponenty byl zcela oddělen od jádra studia, je možné ho využít i při vývoji jiných vizualizačních aplikacích. Do budoucna je plánováno studio rozšířit o další komponenty a dodělat internetové komunikační rozhraní.

PODĚKOVÁNÍ

Tento článek vznikl za podpory interního projektu na podporu studentských vědeckých konferencí SVK-2016-006 a projektu SGS-2015-002: Moderní metody řešení, návrh a aplikace elektronických a komunikačních systémů.

LITERATURA

[1] POKORNÝ, Jan. Řízení a vizualizace robotických vozidel. Plzeň, 2016. Diplomová práce. Západočeská univerzita v Plzni, Fakulta elektrotechnická, Katedra aplikované elektroniky a telekomunikací