Univerzita Hradec Králové

Přírodovědecká fakulta

Katedra fyziky

Komunikační a měřicí software mezi programovatelným můstkem LCR HM8118 a externím zařízením

Bakalářská práce

Autor: Pavel Arnošt

Studijní program: K V13/14 - Fyzikálně-technická měření a výpočetní technika

Studijní obor: Fyzikálně-technická měření a výpočetní technika

Vedoucí práce: Ing. Karol Radocha Ph.D.

Hradec Králové červenec 2020

Zadání:

Cílem zadání je vytvoření software (dále jen programu), který usnadní, zrychlí a zautomatizuje měření pomocí programovatelného můstku LCR HM8118 v laboratořích Přírodovědecké fakulty Univerzity Hradec Králové. Zjednoduší přípravu měřicí aparatury, minimalizuje tvorbu chyb plynoucích z manuální práce při měření a zjednoduší zápis a následné zpracování naměřených hodnot.

K zajištění těchto požadavků je vhodné vytvořit komunikační program, který inicializuje sériové rozhraní RS-232 programovatelného můstku LCR HM8118 od společnosti ROHDE & SCHWARZ a připojí jej k externímu zařízením, čímž je myšleno PC, nebo NB. Běžné používání rozhraní RS-232 je již spíše minulostí, a to pro své méně komfortní použití a malou přenosovou rychlost v porovnání s jinými technologiemi přenosu, nicméně v průmyslovém odvětví, zvláště v oboru měření a regulace má své místo dodnes.

Vzhledem k fyzické absenci tohoto rozhraní v dnešních počítačích je fyzické propojení realizováno pomocí převodníku RS-232 na USB. Dále pomoci vytvořeného GUI bude možné obsluhovat a nastavovat tento můstek, jakož je frekvence, měřící mód, rychlost měření, přesnost měření a následně číst a zobrazovat informace odeslaná můstkem, čímž jsou měřené hodnoty, hodnoty nastavení můstku a takto získaná data uložit pro další možné zpracování.

Program je vytvořen v Microsoft Visual Studio 2019 v jazyce C# využívající platformu .NET Framework verze 3.5.

Prohlášení:

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a že jsem v seznamu použité literatury uvedl všechny prameny, z kterých jsem vycházel.

V Hradci Králové dne 31.7.2020 Pavel Arnošt

Poděkování:

Mé poděkování chci věnovat vedoucímu mé bakalářské práce panu Ing. Karolu Radochovi Ph.D. za trpělivost po celou dobu mého studia, za poskytnutí materiálů a postřehů k vypracování této práce. Taktéž děkuji Přírodovědecké fakultě za vstřícný přístup ke svým studentům. V neposlední řadě chci poděkovat svojí manželce a rodině, za podporu a nekonečnou trpělivost při studiu a psaní závěrečné práce.

Anotace

Účelem této bakalářské práce je vytvořit komunikační program mezi osobním počítačem s měřicím můstkem LCR HM8118 pomocí integrované sériové linky RS-232. Program musí umět vytvořit spojení pomocí sériové linky, odesílat příkazy měřicímu můstku, zpracovávat přijatá data a získaná data uložit ve zpracovatelné formě.

Klíčová slova

Programovatelný měřicí můstek LCR HM8118, sériový port RS-232, převodník RS-232/USB

Annotation

The purpose of this bachelor's thesis is to create a communication program between a personal computer and a measuring bridge LCR HM8118 using an integrated serial port RS-232. The program must be able to establish a connection using a serial port, send commands to the measuring bridge, process the received data and save the obtained data in a processable form.

Keywords

Programmable LCR-Bridge HM8118, serial port RS-232, converter RS232/USB

Obsah:

1 Technické prostředky měřicí soustavy

1.1 Programovatelný můstek LCR HM8118

Programovatelný můstek LCR HM8118 je dvoukanálový laboratorní měřící můstek se schopností měřit elektrické veličiny jako je indukčnost, kapacita, fázový úhel θ, jakost Q, činitel ztrát D, vodivost, odpor, je schopný měřit parametry transformátoru, a to vše s přesností 0,05 %. Můstek má možnost nastavení testovací měřicí frekvence od 20 Hz do 200kHz, klidového napětí (BIAS) jak vnitřní, tak vnější.

Můstek je možné nastavovat z předního panelu obsahujícího ovládací prvky pro nastavování a přepínání měřicích módů, nebo pomocí sériového rozhraní ve dvou variantách, buď připojení skrze RS-232, nebo USB. Výrobce tato rozhraní označuje jako Dual Interface HO820 (USB/RS232). Parametry sériového portu se v obou případech nastavují stejné, je tedy jen na uživateli, který ze dvou způsobů vybere.

V případě použití USB rozhraní můstku je nevyhnutelnou součástí připojení k počítači nutnost provést ručně instalaci USB řadiče můstku dle návodu od dodavatele. Mnohem komfortnější je použití komerčně vyráběného převodníku RS-232 na USB, kdy k instalaci převodníku dojde automaticky bez nutnosti zásahu uživatele. V obou případech počítač ke každému způsobu připojení vytvoří jiný virtuální COM port, tedy označení sériové linky. Teoreticky je možné mezi oběma způsoby připojení přepínat a jen vybírat chtěný COM port.

Obr. 1: Programovatelný můstek LCR HM8118

1.2 Telekomunikační standard IEEE-488 a SCPI

Snaha vyhnout se ručnímu nastavování přístrojů i opisování naměřených hodnot, nebo schopnost toto vše dělat takzvaně na dálku se dá pozorovat od samého počátku rozvoje automatizace. Ovšem až v 70. letech 20. století přichází výrobci na trh s větším množstvím systémů pro tyto účely. Řídicí systém byl s měřicím přístrojem připojen sběrnicí. Tento model zůstal zachován do dnešní doby v mnohých systémech měření a regulace, nejen na komerční úrovni, ale i při domácí tvorbě vlastních návrhů čehokoliv a v neposlední řadě i ve školních laboratořích.

Postupem času s narůstajícím množstvím nejen měřicích zařízení, různých výrobců a různých sběrnic, bylo nevyhnutelné popsat a vytvořit standard, který bude pro svoji univerzálnost použitelný pro širokou škálu řídicích systému i řízených zařízení všech výrobců.

V roce 1975 byl přijat standard IEEE-488 vytvořený pod hlavičkou IEC (International Electrotechnical Commission), který definuje mechanicko-elektrické části propojených přístrojů a základní protokoly, dá se říct, definuje fyzickou až síťovou vrstvu. Neexistence standardizace syntaxe příkazů, způsobu formátování dat pochopitelně vedla k další standardizaci těchto vyšších vrstev. V roce 1978 tedy vzniká IEEE-488.2 - Standard Codes, Formats, Protocols, and Common Commands. Tento standard měl původně využívat výhradně již existující sběrnici GPIB (General Purpose Interface Bus).

Poslední standardizací prošla nejvyšší vrstva definující příkazy kompatibilní mezi všemi měřicími zařízeními tedy SCPI (Standard Commands for Programmable Instruments). SCPI je tedy jednoduchý jazyk pro řízení měřicích zařízení.

Při dalším vývoji počítačové i měřící techniky došlo k použití standardu IEEE-488.2 a SCPI i na jiné rozhraní, např. RS-232, Ethernet atd.

1.3 SCPI u HM8118

SCPI standard definuje konkrétní příkazy, formát i strukturu příkazů pro měřicí zařízení. Ovšem výrobce HM8118 vytvořil vlastní sadu příkazů pro ovládání svých zařízení. Pro HM8118 se jedná textové příkazy o délce 4 znaků, které jsou

pomyslnou zkratkou svého nezkráceného znění, např. příkaz FREQ pracující s testovací frekvencí můstku je zkratkou z původního slova Frequency.

Příkazy zle rozdělit na dva druhy. Následuje-li za textem příkazu otazník (?) jedná se o dotaz na nastavení můstku a očekává se od můstku odpověď v kódování ASCII, např. na dotaz FREQ? obdržíme číselnou hodnotu vyjadřující hodnotu testovací frekvence v Hz. Pokud však za textem příkazu následuje číselná hodnota, jedná se o příkaz k nastavení stavu můstku, např. FREQ100 nastaví testovací frekvenci můstku na hodnotu 100 Hz.

Pro správnou inicializaci i funkci sériové komunikace mezi řídicím systémem a měřicím můstkem je vyžadováno na konci každého jednotlivě volaného příkazu odeslat netisknutelný řídicí znak (\r), neboli CR (Carriage Return). Tento znak obvykle vrací kurzor na začátek řádku. Zde je jeho význam rozšířen o funkci jednoznačného ukončení volané sekvence. Kompletní příkaz vypadá takto: FREQ100\r.

Tento netisknutelný znak je zasílán měřicím můstkem řídicímu systému i v případě předchozího dotazu řídicího systému. Znak ukončuje každou zaslanou odpověď, např. na dotaz FREQ?\r je možnou odpovědí 1000\r.

Z výše uvedeného tedy vyplývá, že byť jsou příkazy SCPI považovány za programovací jazyk, tak samotný řídicí systém je nezná a zaslaným odpovědím nerozumí. Je na samotném programátorovi, nebo uživateli porozumět příkazům a odpovědím a tím řídit měřicí můstek. Tato zdánlivá nevýhoda se však otáčí na výhodu pro svoji nezávislost na jediném programovacím jazyku.

1.4 Sériový port RS-232

Výraz RS-232 má původ v telekomunikaci označující standard pro sériovou komunikaci dat mezi DTE (Data Terminal Equipment – např. osobní počítač) a DCE (Data Circuic Terminating Equipmnet – např. modem). Standard RS-232 definuje elektrické charakteristiky sériové komunikace, časování signálů i tvar a rozměry konektorů. Sériová komunikace znamená, že jednotlivé bity jsou přenášeny

samostatně za sebou, pomocí jednoho páru vodičů pro každý směr komunikace. Dva páry vodičů umožňují zcela bezkolizní fyzickou vrstvu.

Standard definuje dvě logické hodnoty log.0 a log.1. Logická 1 odpovídá stejnosměrnému napětí -12 V, pro logickou 0 je napětí 12 V. Standard udává meze logických napěťových úrovní vysílače i přijímače. U vysílače jsou meze logické 0 od +5 V do + 15 V, logická 1 má meze napětí od -15 V do -5 V. Přijímač má meze mnohem širší, logickou 0 rozezná mezi +3 až +25 V, logickou 1 od -25 V do -3 V. Toto platí pro PINy TxD a RxD, u ostatních pinů jsou napěťové úrovně opačné, jinak řečeno mají obrácenou logiku.

Obr. 2: Konektor sériového portu RS-232

Základní schéma zapojení obsahuje 9 PINů, neboli vodičů. Základní vodiče pro vysílání a příjem (TxD a RxD) doplňuje společná zem (GND). Další vodiče (RTS, CTS, DSR, DCD, DTR, RI) slouží k obsluze přenosu.

TxD	Transmit Data	Přenos dat z DTE do DCE
RxD	Receive Data	Příjem dat v DTE z DCE
GND	Ground	Signálová zem

RTS	Request to Send	Nastavením logické 1 DTE signalizuje požadavek na vysílání
CTS	Clear to Send	Logickou 1 povoluje DCE okamžité vysílání dat z DTE do DCE
DSR	Data Set Ready	Logickou 1 oznamuje DCE připravenost pro příjem dat, který může začít až po nastavení CTS na logickou 1
DCD	Data Carrier Detect	Nastavením logické 1 DCE signalizuje detekci signálu a je připraveno komunikovat
DTR	Data Terminal Ready	DTE nastavením logické 1 signalizuje DCE připravenost pro přenos dat, DCE následně nastavuje DSR na logickou 1
RI	Ring Indicator	DCE nastaví logickou 1 a signalizuje do DTE, že DCE žádá Datové spojení

Tab. 1: Názvy PINů sériového portu RS-232

Rychlost přenosu je dle standardu maximálně 115200 Bd (Bd je jednotka počtu změn napěťových úrovní za vteřinu). Dále se číselná řada dělí na poloviny. Mezi nejběžnější používané přenosové rychlosti patří 115200 Bd, 57600 Bd, 38400 Bd, 19200 Bd, 9600 Bd, 4800 Bd a 2400 Bd. Jelikož se při realizaci přenosu používají i jiné bity než datové, je výsledná přenosová rychlost vždy nižší, než na jakou jsou DTE i DCE nastaveny.

Samotný přenos jednotlivých bitů probíhá od LSB (Least Significant Bit – nejméně významný bit) po MSB (Most Significant Bit – nejvýznamnější bit).

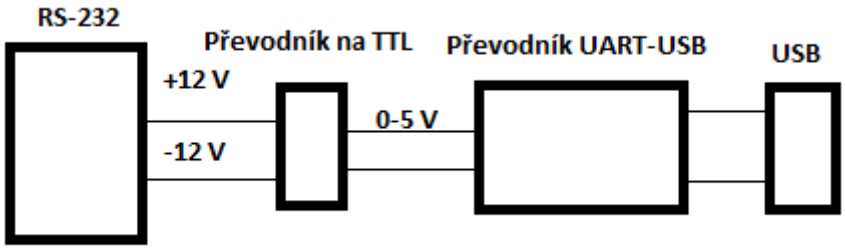
Jak již bylo zmíněno, přenosová kapacita je vždy nižší, než je přenosová rychlost sériového portu. To souvisí s principem asynchronního přenosu, který standard RS-232 využívá, totiž k synchronizaci přenosu mezi DTE a DCE je třeba zaslat u tohoto standardu start bit. Následně dochází k přenosu vlastních dat. Po vlastních datech následuje paritní bit, sloužící jako jednoduché ověření správnosti zaslaných dat, a nakonec jeden nebo více stop bitů.

1.5 Převodník RS-232 na USB

Nezbytnou součástí řešení je využití převodníku z RS-232 na USB. Jak bylo již popsáno, uživatel může využít jeden ze dvou výstupů Dual Interface HO820. Jednou variantou je využití výstupu USB s typem konektoru USB B, druhou variantou je přímý výstup RS-232. V obou případech se technicky využívá převodníku z RS-232

na USB. Důvodem nutného využití převodníku je v rozdílných napětích mezi RS-232 a USB. Sériová linka RS-232 obvykle využívá rozsah ±12 V, kdežto USB má rozsah 0 až 5 V.

Obecně lze říci, že se tento převodník skládá ze dvou částí, dvou integrovaných obvodů. První obvod je buď obvod MAX232, nebo jeho odvozenina. Dokáže převést signály napěťové úrovně ±12 V na úroveň TTL (Transistor Transistor Logic – tranzistově-tranzistorová logika) v obou směrech. Druhým integrovaným obvodem je samotný převodník USB-UART (Universal Asynchronous Receiver Transmitter), obvykle se jedná o řadu FT232. Tento integrovaný obvod již dokáže převést způsob komunikace po dvou linkách u RS-232 na linku jednu u USB.



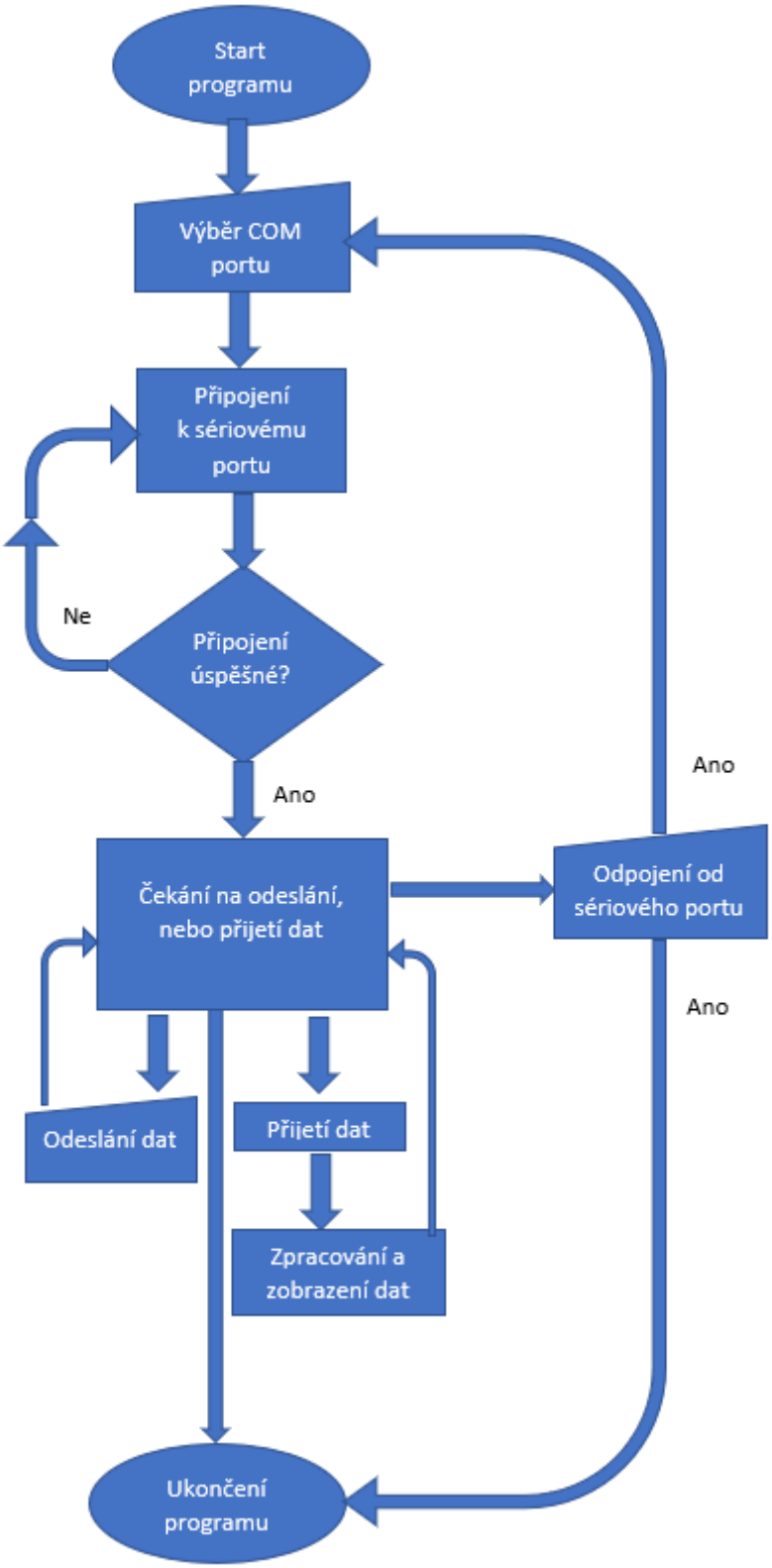
Obr. 3: Schéma převodníku RS-232 na USB

2 Vytvoření komunikačního programu s HM8118

Řešení programu je založeno na využití třídy SerialPort programovacího jazyka C# s podporou .NET verze 3.5. Tato třída obsahuje všechny potřebné prostředky pro navázání a řízení sériové komunikace mezi DTE a DCE. V manuálu k HM8118 je uvedeno konkrétní nastavení, jež zaručí spojení mezi řídicím systémem a měřicím můstkem. Konkrétně se v manuálu uvádí nastavení přenosové rychlosti na 9600 Bd, 8 datových bitů, 1 stop bit bez použití parity. Tyto hodnoty jsou tedy do programu nastaveny natvrdo bez možnosti změny uživatelem.

Program je vytvořen jako spustitelná aplikace Windows Form s obslužnými prvky.

2.1 Vývojový diagram



Obr. 4: Vývojový diagram komunikačního programu

2.2 GUI aplikace

Komunikační program s můstkem HM8118 jsem navrhnul jako spustitelnou aplikaci, s jednoduchým ovládáním. Aplikace využívá jediné okno ke všem potřebným úkonům, jako je výběr COM portu, připojení, či odpojení vybraného COM portu, zobrazení korektního připojení, textová pole pro vstupní příkazy i odpovědi měřicího můstku, tlačítka pro odeslání příkazů, pro výmaz odeslaných, či přijatých řetězců a tlačítka pro nahrání či uložení vstupních, nebo výstupních dat.

Obr. 5 GUI aplikace komunikačního programu

2.3 Spuštění aplikace

Spuštění aplikace se provádí spustitelným souborem HM8118.exe. Následně se otevře GUI aplikace. Ta na pozadí načte v tu chvíli dostupné COM porty. Ty vloží do rozbalovacího seznamu pojmenovaného COM. V tento moment není uživateli umožněno odesílat data skrz sériový port, nebo tento port odpojit, jelikož doposud nedošlo ke korektnímu připojení k vybranému sériovému portu uživatelem. Omezení je zařízeno neaktivními tlačítky pro odpojení sériového portu a tlačítky pro odesílání dat skrz sériový port.

2.4 Výběr COM portu

Jak bylo uvedeno, po spuštění aplikace se do rozbalovacího seznamu COM vloží seznam aktuálně dostupných, tedy připojitelných portů COM v počítači. Je-li do seznamu vloženo více připojitelných COM portů, není na první pohled možné určit který COM port (např. COM2) patří právě měřicímu můstku. Zjištění, či případně

změna čísla COM portu je možná dle způsobu použití Dual Interface HO820. U USB B výstupu, je po instalaci možné ve Správci zařízení v počítači vyčíst, nebo změnit číslo COM portu, pod kterým se aktuálně tento výstup mapuje. Při použití komerčního převodníku z výstupu RS-232 měřicího můstku záleží na vlastnostech pořízeného převodníku. Jelikož způsobů zjištění i úprav je tolik co výrobců převodníků RS-232 na USB, není vhodné všechny možnosti popisovat do této práce, ani to není jejím úkolem.

2.5 Připojení a odpojení sériového portu

Připojení k vybranému sériovému portu se provede stiskem tlačítka CONNECT. Není-li vybrán žádný COM port ze seznamu, objeví se upozorňující zpráva na tuto skutečnost. Úspěšnost připojení se projeví změnou barvy Status Baru na zelenou a aktivováním tlačítek pro odesílání dat směrem k měřicímu můstku.

Pro odpojení sériového portu je možné stisknout tlačítko DISCONNECT. To vyvolá korektní odpojení portu a deaktivaci tlačítek pro odesílání dat a tím zajistí ochranu proti pokusu o odeslání dat sériovým portem..

2.6 Vstupní pole

Vstupní pole je realizováno komponentou TextBox. Jedná se o objekt, do kterého je možné zapisovat textový vstup z klávesnice nejen řídicího systému. Umožňuje víceřádkový zápis oddělený znakovým oddělovačem. Toto pole je pro uživatele zcela otevřené, je mu umožněn zápis, výmaz i editace. Maximální délka vstupního textu není nijak omezena.

2.7 Odesílání dat

Pro odesílání dat jsou vytvořena dvě tlačítka pro komfortnější práci s aplikací. Tlačítko SEND odešle zapsaná data uživatele ve vstupním TextBoxu, tlačítko SEND and CLEAR vstupní data taktéž odešle, a navíc obsah vstupního TextBoxu vymaže. Ve skutečnosti se vždy volá stejná vytvořená metoda, jednou bez příznaku pro výmaz vstupního TextBoxu, ve druhém případě s příznakem pro výmaz obsahu TextBoxu.

. Všechny zapsané příkazy ve vstupním TextBoxu se nejprve rozdělí na jednotlivé příkazy, uloží se do pomocného pole stringů a následně se využitím metody WriteLine ze třídy SerialPort odešlou sériovým portem včetně zakončovacího znaku \r. Rozdělené příkazy jsou uloženy do pomocného pole stringů. Tento postup není nutný, samotný měřicí můstek jej nevyžaduje. Měřicí můstek umí zpracovat více příkazů za sebou, jen musí být dodržena konvence zakončení znakem \r. Pro tento postup jsem se rozhodl pro komfortnější přehled obsluhy nad celkovou komunikací a pro možnost exportu komunikace včetně naměřených dat. Výše již bylo popsáno, že na dotazový příkaz např. FREQ? odpoví můstek hodnotou např. 1000, nikoliv sdělením FREQ=1000, tudíž při odeslání více dotazů může obsluha snadno ztratit přehled, k jakému dotazu se tato odpověď vztahuje. Připomeňme, že komunikace neprobíhá synchronně, tedy že odpověď zpravidla přijde v době, kdy byl můstku zaslán již jiný příkaz, nebo dotaz.

2.8 Mazání dat vstupního pole

V praxi jistě nastanou situace, kdy obsluha bude chtít vymazat všechny zapsané příkazy ve vstupním TextBoxu. To může provést několika způsoby. Použitím tlačítka SEND and CLEAR, ručním výmazem zapsaného textu, nebo připraveným tlačítkem CLEAR. Stisknutí toho tlačítka zajistí kompletní vymazaní vstupního TextBoxu a nastavení kurzoru na jeho začátek.

2.9 Načtení dat do vstupního pole

Zadání práce obsahuje požadavek na možnost uložení naměřených dat v dále zpracovatelné formě. Pochopitelně pro úplnost je aplikace doplněna i o možnost načtení dat. Tímto se rozumí logický sled příkazů a dotazů určených pro měřicí můstek. Soubor s tímto prostým seznamem příkazů se očekává ve formátu *.txt. K samotnému otevírání souborů je využita třída OpenFileDialog. Ta po stisku tlačítka LOAD nabídne uživateli standardní okno pro otevírání souborů. Uživatel tedy může libovolně procházet souborovou strukturu pro nalezení vhodného souboru s příkazy ve formátu *.txt. Dialog otevírání souboru má přednastaveno zobrazování a otevírání souborů právě s příponou *.txt. Po korektním výběru a otevření souboru

je jeho obsah vložen do vstupního TextBoxu. Funkce načtení dat ze souboru není podmíněna aktivním připojením sériového portu.

2.10 Výstupní pole

Výstupní pole je realizována taktéž komponentou TextBox. Data jsou do TextBoxu zapisována včetně vstupní hodnoty zasílané do měřicího můstku. Data vstupní i data výstupní jsou oddělena formátovacím znakem „;“.

Do výstupního TextBoxu se data doplňují pomocí metody ReadExisting třídy SerialPort, ta zajišťuje asynchronní čtení sériového portu a následný výpis přijatých dat do výstupního TextBoxu. Do výstupního TextBoxu je uživateli dovoleno zapisovat dle libosti, například pro zápis poznámek spojených s měřicí úlohou.

2.11 Výmaz výstupního pole

Uživatel má taktéž možnost vymazat obsah výstupního pole stiskem tlačítka CLEAR u výstupního TextBoxu.

2.12 Uložení obsahu výstupního pole do souboru

Uložení naměřených dat k dalšímu zpracování je jedním z bodu zadání práce. Podobně jako u načtení souboru se zde využívá třída SaveFileDialog. Ta vyvolá standardní okno pro uložení souboru. Uložení do souboru se skrývá pod tlačítkem SAVE. To mimo vyvolání nabídky k uložení dat do souboru omezí uživatele na jediný možný výstupní formát, a to *.txt. Data z výstupních TextBoxů na stejných řádcích oddělí formátovacím znakem „;“.

2.13 Uzavření aplikace

Uzavření spuštěné aplikace se provádí standardním způsobem pro WindowsForm aplikace, tedy křížkem vpravo nahoře spuštěné aplikace. Takovéto ukončení aplikace nezpůsobí jen ukončení aplikace samotné, ale taktéž ukončí existující spojení mezi počítačem uživatele a měřicím můstkem.

3 Závěr

V praxi i školním prostředí se často setkáváme s úlohami měření vlastností elektronických součástek, časových průběhů elektrického napětí, či proudů, nebo mnohem složitějších úloh. Všechny tyto úlohy mají v principu mnoho společné. Připravení měřicí úlohy, provedení samotného měření a zápis naměřených hodnot. Od samého počátku měřicích úloh napříč obory existuje snaha o automatizaci těchto kroků včetně zpracování získaných údajů. Odstranění chybovosti při ručním zápisu naměřených hodnot je jedním z možností, jak nejen celý proces urychlit, ale i zvýšit věrohodnost zapsaných dat, potažmo naměřená data mít k dispozici pro jednodušší následné zpracování a vyhodnocení. V neposlední řadě spolehnutí se na proces získání naměřených dat a jejich záznam otevírá větší možnost pro uživatele se soustředit na samotný proces přípravy měření, nebo jeho vyhodnocení.

To byl i záměr této bakalářské práce. Vytvořit program, který zajistí komunikaci s měřicím můstkem s řídicím systémem. Měřicí můstek nabízí komunikaci skrze sériový port RS-232. Dále se v manuálu dá zjistit konkrétní nastavení pro sériovou komunikaci a všechny příkazy pro ovládání a nastavování můstku.

Výsledkem zadání je program, který naváže komunikaci s měřicím můstkem, má vstupní i výstupní pole pro zápis řídicích příkazů, potažmo pro zobrazení odeslaných dat z můstku. Dále umožňuje načtení řídicích příkazů ze souboru a taktéž uložení do souboru dat z měření.

4 Zdroje

1) RS232. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-07-31]. Dostupné z: https://cs.wikipedia.org/wiki/RS-232

2) RS232. Papouch.com [online]. [cit. 2020-07-31]. Dostupné z: https://papouch.com/seriovy-port-rs232-p3740/

3) Dokumentace pro .NET. Microsof Doc [online]. 2020 [cit. 2020-07-31]. Dostupné z: https://docs.microsoft.com/dotnet/

4) Diplomová práce Aplikace pro dálkové ovládání přístrojů Rohde & Schwarz. Dopravní fakulta Jana Pernera, 2013. Diplomová práce. Univerzita Pardubice. Vedoucí práce Ing. Jan Pinadič.

5) Programmable LCR-Bridge HM8118: Manual. 12.04.2012. Mainhausen: HAMEG Instruments, 2012. ISBN 45-8118-0010.

Obrázky:

6) Programovatelný můstek LCR HM8118. In Programmable LCR-Bridge HM8118: Manual. 12.04.2012. Mainhausen: HAMEG Instruments, 2012. ISBN 45-8118-0010. – Obr. 1

7) Konektor sériového portu RS-232. In: Wikipedia [online]. [cit. 2020-07-31]. Dostupné z: https://cs.wikipedia.org/wiki/RS-232#/media/Soubor:9_pin_d-sub_connector_male_closeup.jpg – Obr. 2

5 Přílohy

5.1 Zdrojový kód fHAMEG.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.IO;

namespace verze01
{
    public partial class HAMEG : Form
    {
        // Deklarace globálních proměnných
        string[] receiveIN = new string[] {};
        string[] receiveOUT = new string[] {};
        string dataIN;
        int count = 0;
        int count1 = 0;
        int count2 = 0;

        public HAMEG()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // Aktivace tlačítka pro připojení sériového portu
            bCONNECT.Enabled = true;

            // Deaktivace tlačítek pro odesílání dat sériovým portem
            bDISCONNECT.Enabled = false;
            bSEND.Enabled = false;
            bSENDandCLEAR.Enabled = false;

            // Načtení dostupných COM portů a vložení do rozbalovacího menu
            string[] ports = SerialPort.GetPortNames();
            cBoxCOM.Items.AddRange(ports);
        }

        private void groupBox1_Enter(object sender, EventArgs e)
        {

        }

        private void bDISCONNECT_Click(object sender, EventArgs e)
        {
            // Metoda pro ukončení komunikace sériovým portem

            // Ověření zda je sériový port otevřen
            if (sPort.IsOpen)
```

```
            {
                sPort.Close();
                pBAR.Value = 0;

                // Aktivace tlačítka pro připojení sériového portu, deaktivace
tlačítek pro odesílání dat
                bCONNECT.Enabled = true;
                bDISCONNECT.Enabled = false;
                bSEND.Enabled = false;
                bSENDandCLEAR.Enabled = false;
            }
        }

        private void bCONNECT_Click(object sender, EventArgs e)
        {
            // Metoda pro otevření sériového portu
            try
            {
                // Pokus o otevření sériového portu

                // Nastavení parametrů sériového portu dle manuálu výrobce
                sPort.PortName = cBoxCOM.Text;
                sPort.Parity = Parity.None;
                sPort.BaudRate = 9600;
                sPort.DataBits = 8;
                sPort.StopBits = StopBits.One;
                sPort.RtsEnable = false;

                sPort.Open();
                pBAR.Value = 100;

                // Deaktivace tlačítka pro otevření portu
                bCONNECT.Enabled = false;

                // Aktivace tlačítek pro odesílaní dat
                bDISCONNECT.Enabled = true;
                bSEND.Enabled = true;
                bSENDandCLEAR.Enabled = true;
            }
            catch (Exception err)
            {
                // Zachycení výjimky a zobrazení chybové hlášky v případě
neúspěšného pokusu o otevření portu
                MessageBox.Show(err.Message,"Chyba", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void bSEND_Click(object sender, EventArgs e)
        {
            // Volání vlastní metody pro odesílání dat bez výmazu obsahu
vstupního TextBoxu
            SEND(false);
        }

        private void groupBox2_Enter(object sender, EventArgs e)
        {

        }

        private void sPort_DataReceived(object sender,
SerialDataReceivedEventArgs e)
```

```
        {

            // Metoda pro příjem a zpracování dat

            // Nastavení TimeOutu pro zajištění příjmu dat až po odeslání
            System.Threading.Thread.Sleep(400);

            // Načtení dat do proměnné dataIN
            dataIN = sPort.ReadExisting();

            // Rozdělení vstupního řetězce dle formátovacích znaků
            string[] stringSeparators = new string[] { "\r", " "};
            string[] result;
            result = dataIN.Split(stringSeparators,
StringSplitOptions.RemoveEmptyEntries);


            // Pomocná počítadla
            count1 = 0;
            count2 = 0;

            // Zápis přijatých dat na pozice v poli receiveOUT kde je zapsán
pomocný znak x
            foreach (string entry in receiveOUT)
            {
                    if (receiveOUT[count1].Equals("x"))
                    {
                        receiveOUT[count1] = result[count2];
                        count2++;
                    }
                    count1++;
                if (count2 >= result.Length)
                { break; }
            }

            this.Invoke(new EventHandler(ShowData));
        }

        private void ShowData(object sender, EventArgs e)
        {
            // Přiřazení výstupních stringů příslušným vstupním, vložení
středníku a odřádkování
            for (int i = 0; i < receiveIN.Length; i++)
            {
                tBoxRECEIVE.Text += (receiveIN[i] + ";" + receiveOUT[i]);
                tBoxRECEIVE.Text += "\r\n";
            }
        }

        private void bSENDandCLEAR_Click(object sender, EventArgs e)
        {
            // Volání metody pro odeslání dat s příznakem na výmaz vstupního
TextBoxu
            SEND(true);
            tBoxSEND.Text = "";
        }

        private void bSENDclear_Click(object sender, EventArgs e)
        {
            // Výmaz vstupního TextBoxu
            tBoxSEND.Text = "";
        }
```

```
        private void bRECEIVEclear_Click(object sender, EventArgs e)
        {
            // Výmaz výstupního TextBoxu i pomocných polí
            tBoxRECEIVE.Text = "";
            Array.Resize(ref receiveOUT, 0);
            Array.Resize(ref receiveIN, 0);
        }

        private void bSAVE_Click(object sender, EventArgs e)
        {
            // Metoda pro uložení dat z výstupního TextBoxu do souboru formátu
*.txt
            Stream myStream;
            SaveFileDialog saveFileDialog1 = new SaveFileDialog();

            saveFileDialog1.Filter = "txt files(*.txt) | *.txt | All files(*.*) |
*.* ";
            saveFileDialog1.FilterIndex = 1;
            saveFileDialog1.RestoreDirectory = true;

            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                if ((myStream = saveFileDialog1.OpenFile()) != null)
                {
                    StreamWriter streamWriter = new StreamWriter(myStream);
                    streamWriter.Write(tBoxRECEIVE.Text);
                    streamWriter.Close();
                    myStream.Close();
                }
            }
        }

        private void bLOAD_Click(object sender, EventArgs e)
        {
            // Metoda pro otevření souboru a vložení jeho obsahu do vstupního
TextBoxu
            var fileContent = string.Empty;
            var filePath = string.Empty;

            using (OpenFileDialog openFileDialog = new OpenFileDialog())
            {
                openFileDialog.InitialDirectory = "c:\\";
                openFileDialog.Filter = "txt files(*.txt) | *.txt | All
files(*.*) | *.* ";
                openFileDialog.FilterIndex = 1;
                openFileDialog.RestoreDirectory = true;

                if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    //Get the path of specified file
                    filePath = openFileDialog.FileName;

                    //Read the contents of the file into a stream
                    var fileStream = openFileDialog.OpenFile();

                    using (StreamReader reader = new StreamReader(fileStream))
                    {
                        tBoxSEND.Text = reader.ReadToEnd();
                    }
                }
            }
```

```
        }
        public void SEND(bool clear)
        {
            // Vlastní metoda k odesílání dat sériovým portem

            // Ověření zda je s=riový port stále otevřen a zda je vstupní TextBox
neprázdný
            if (sPort.IsOpen && tBoxSEND.Text != "")
            {
                // Rozdělení textu vstupního TextBoxu na jednotlivé příkazy dle
formátovacích znaků
                string[] stringSeparators = new string[] { "\r\n" };
                string[] result;
                result = tBoxSEND.Text.Split(stringSeparators,
StringSplitOptions.RemoveEmptyEntries);

                // Změna velikostí pomocných polí receiveOut a receiveIN dle
počtu zadaných příkazů
                Array.Resize(ref receiveOUT, result.Length + receiveIN.Length);
                Array.Resize(ref receiveIN, result.Length + receiveIN.Length);

                // Každý dílčí příkaz je volán s povinným znakem \r\n
                foreach (string entry in result)
                    {
                        sPort.WriteLine(entry + "\r\n");

                    // Jedná-li se o dotazový příkaz jenž končí otazníkem, tak na
stejný řádek výstupního pomocného pole stringů vložím symbol x, jinak vkládám -
                    receiveIN[count] = entry;
                    if (entry[(entry.Length - 1)].Equals('?'))
                    {
                        receiveOUT[count] = "x";
                    }
                    else receiveOUT[count] = "-";
                    count++;
                    }

                // Sekvence pro výmaz pomocného vstupního pole změnou velikosti
pole na 0
                if (clear)
                {
                    Array.Resize(ref result, 0);
                }

            }
        }
    }
}
```

5.2 Zdrojový kód fHAMEG.Designer.cs

```
namespace verze01
{
    partial class HAMEG
    {
        /// <summary>
        /// Vyžaduje se proměnná návrháře.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
```

```
        /// Uvolněte všechny používané prostředky.
        /// </summary>
        /// <param name="disposing">hodnota true, když by se měl spravovaný
prostředek odstranit; jinak false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Kód generovaný Návrhářem Windows Form

        /// <summary>
        /// Metoda vyžadovaná pro podporu Návrháře - neupravovat
        /// obsah této metody v editoru kódu.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.pBAR = new System.Windows.Forms.ProgressBar();
            this.bDISCONNECT = new System.Windows.Forms.Button();
            this.bCONNECT = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.cBoxCOM = new System.Windows.Forms.ComboBox();
            this.tBoxSEND = new System.Windows.Forms.TextBox();
            this.bSEND = new System.Windows.Forms.Button();
            this.sPort = new System.IO.Ports.SerialPort(this.components);
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.bSENDclear = new System.Windows.Forms.Button();
            this.bSENDandCLEAR = new System.Windows.Forms.Button();
            this.groupBox3 = new System.Windows.Forms.GroupBox();
            this.bSAVE = new System.Windows.Forms.Button();
            this.bRECEIVEclear = new System.Windows.Forms.Button();
            this.tBoxRECEIVE = new System.Windows.Forms.TextBox();
            this.bLOAD = new System.Windows.Forms.Button();
            this.groupBox1.SuspendLayout();
            this.groupBox2.SuspendLayout();
            this.groupBox3.SuspendLayout();
            this.SuspendLayout();
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.pBAR);
            this.groupBox1.Controls.Add(this.bDISCONNECT);
            this.groupBox1.Controls.Add(this.bCONNECT);
            this.groupBox1.Controls.Add(this.label1);
            this.groupBox1.Controls.Add(this.cBoxCOM);
            this.groupBox1.Location = new System.Drawing.Point(13, 13);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(200, 156);
            this.groupBox1.TabIndex = 0;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Com Port Select";
            this.groupBox1.Enter += new
System.EventHandler(this.groupBox1_Enter);
            //
            // pBAR
            //
```

```
            this.pBAR.Location = new System.Drawing.Point(19, 111);
            this.pBAR.Name = "pBAR";
            this.pBAR.Size = new System.Drawing.Size(175, 23);
            this.pBAR.TabIndex = 4;
            //
            // bDISCONNECT
            //
            this.bDISCONNECT.Location = new System.Drawing.Point(19, 81);
            this.bDISCONNECT.Name = "bDISCONNECT";
            this.bDISCONNECT.Size = new System.Drawing.Size(175, 23);
            this.bDISCONNECT.TabIndex = 3;
            this.bDISCONNECT.Text = "DISCONNECT";
            this.bDISCONNECT.UseVisualStyleBackColor = true;
            this.bDISCONNECT.Click += new
System.EventHandler(this.bDISCONNECT_Click);
            //
            // bCONNECT
            //
            this.bCONNECT.Location = new System.Drawing.Point(19, 52);
            this.bCONNECT.Name = "bCONNECT";
            this.bCONNECT.Size = new System.Drawing.Size(175, 23);
            this.bCONNECT.TabIndex = 2;
            this.bCONNECT.Text = "CONNECT";
            this.bCONNECT.UseVisualStyleBackColor = true;
            this.bCONNECT.Click += new System.EventHandler(this.bCONNECT_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(16, 22);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(31, 13);
            this.label1.TabIndex = 1;
            this.label1.Text = "COM";
            //
            // cBoxCOM
            //
            this.cBoxCOM.FormattingEnabled = true;
            this.cBoxCOM.Location = new System.Drawing.Point(73, 19);
            this.cBoxCOM.Name = "cBoxCOM";
            this.cBoxCOM.Size = new System.Drawing.Size(121, 21);
            this.cBoxCOM.TabIndex = 0;
            //
            // tBoxSEND
            //
            this.tBoxSEND.Location = new System.Drawing.Point(6, 19);
            this.tBoxSEND.Multiline = true;
            this.tBoxSEND.Name = "tBoxSEND";
            this.tBoxSEND.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;
            this.tBoxSEND.Size = new System.Drawing.Size(121, 153);
            this.tBoxSEND.TabIndex = 1;
            //
            // bSEND
            //
            this.bSEND.Location = new System.Drawing.Point(133, 17);
            this.bSEND.Name = "bSEND";
            this.bSEND.Size = new System.Drawing.Size(78, 23);
            this.bSEND.TabIndex = 2;
            this.bSEND.Text = "SEND";
            this.bSEND.UseVisualStyleBackColor = true;
            this.bSEND.Click += new System.EventHandler(this.bSEND_Click);
            //
```

```
            // sPort
            //
            this.sPort.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(this.sPort_DataReceived);
            //
            // groupBox2
            //
            this.groupBox2.Controls.Add(this.bLOAD);
            this.groupBox2.Controls.Add(this.bSENDclear);
            this.groupBox2.Controls.Add(this.bSENDandCLEAR);
            this.groupBox2.Controls.Add(this.tBoxSEND);
            this.groupBox2.Controls.Add(this.bSEND);
            this.groupBox2.Location = new System.Drawing.Point(219, 13);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(217, 185);
            this.groupBox2.TabIndex = 3;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Send";
            this.groupBox2.Enter += new
System.EventHandler(this.groupBox2_Enter);
            //
            // bSENDclear
            //
            this.bSENDclear.Location = new System.Drawing.Point(133, 81);
            this.bSENDclear.Name = "bSENDclear";
            this.bSENDclear.Size = new System.Drawing.Size(78, 23);
            this.bSENDclear.TabIndex = 4;
            this.bSENDclear.Text = "CLEAR";
            this.bSENDclear.UseVisualStyleBackColor = true;
            this.bSENDclear.Click += new
System.EventHandler(this.bSENDclear_Click);
            //
            // bSENDandCLEAR
            //
            this.bSENDandCLEAR.Location = new System.Drawing.Point(133, 46);
            this.bSENDandCLEAR.Name = "bSENDandCLEAR";
            this.bSENDandCLEAR.Size = new System.Drawing.Size(78, 23);
            this.bSENDandCLEAR.TabIndex = 3;
            this.bSENDandCLEAR.Text = "SEND and CLEAR";
            this.bSENDandCLEAR.UseVisualStyleBackColor = true;
            this.bSENDandCLEAR.Click += new
System.EventHandler(this.bSENDandCLEAR_Click);
            //
            // groupBox3
            //
            this.groupBox3.Controls.Add(this.bSAVE);
            this.groupBox3.Controls.Add(this.bRECEIVEclear);
            this.groupBox3.Controls.Add(this.tBoxRECEIVE);
            this.groupBox3.Location = new System.Drawing.Point(452, 13);
            this.groupBox3.Name = "groupBox3";
            this.groupBox3.Size = new System.Drawing.Size(289, 185);
            this.groupBox3.TabIndex = 4;
            this.groupBox3.TabStop = false;
            this.groupBox3.Text = "Receive";
            //
            // bSAVE
            //
            this.bSAVE.Location = new System.Drawing.Point(199, 128);
            this.bSAVE.Name = "bSAVE";
            this.bSAVE.Size = new System.Drawing.Size(78, 50);
            this.bSAVE.TabIndex = 6;
            this.bSAVE.Text = "SAVE";
```

```
            this.bSAVE.UseVisualStyleBackColor = true;
            this.bSAVE.Click += new System.EventHandler(this.bSAVE_Click);
            //
            // bRECEIVEclear
            //
            this.bRECEIVEclear.Location = new System.Drawing.Point(200, 19);
            this.bRECEIVEclear.Name = "bRECEIVEclear";
            this.bRECEIVEclear.Size = new System.Drawing.Size(78, 50);
            this.bRECEIVEclear.TabIndex = 5;
            this.bRECEIVEclear.Text = "CLEAR";
            this.bRECEIVEclear.UseVisualStyleBackColor = true;
            this.bRECEIVEclear.Click += new
System.EventHandler(this.bRECEIVEclear_Click);
            //
            // tBoxRECEIVE
            //
            this.tBoxRECEIVE.Location = new System.Drawing.Point(7, 19);
            this.tBoxRECEIVE.Multiline = true;
            this.tBoxRECEIVE.Name = "tBoxRECEIVE";
            this.tBoxRECEIVE.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
            this.tBoxRECEIVE.Size = new System.Drawing.Size(187, 160);
            this.tBoxRECEIVE.TabIndex = 0;
            //
            // bLOAD
            //
            this.bLOAD.Location = new System.Drawing.Point(133, 122);
            this.bLOAD.Name = "bLOAD";
            this.bLOAD.Size = new System.Drawing.Size(78, 50);
            this.bLOAD.TabIndex = 7;
            this.bLOAD.Text = "LOAD";
            this.bLOAD.UseVisualStyleBackColor = true;
            this.bLOAD.Click += new System.EventHandler(this.bLOAD_Click);
            //
            // HAMEG
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(767, 235);
            this.Controls.Add(this.groupBox3);
            this.Controls.Add(this.groupBox2);
            this.Controls.Add(this.groupBox1);
            this.Name = "HAMEG";
            this.Text = "HAMEG";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.groupBox2.ResumeLayout(false);
            this.groupBox2.PerformLayout();
            this.groupBox3.ResumeLayout(false);
            this.groupBox3.PerformLayout();
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.ComboBox cBoxCOM;
        private System.Windows.Forms.ProgressBar pBAR;
        private System.Windows.Forms.Button bDISCONNECT;
```

```
        private System.Windows.Forms.Button bCONNECT;
        private System.Windows.Forms.TextBox tBoxSEND;
        private System.Windows.Forms.Button bSEND;
        private System.IO.Ports.SerialPort sPort;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.TextBox tBoxRECEIVE;
        private System.Windows.Forms.Button bSENDandCLEAR;
        private System.Windows.Forms.Button bSENDclear;
        private System.Windows.Forms.Button bRECEIVEclear;
        private System.Windows.Forms.Button bSAVE;
        private System.Windows.Forms.Button bLOAD;
    }
}
```