UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

JEDNODUCHÉ HMI VYBAVENÍ PRO MONITOROVÁNÍ REÁLNÉHO PROCESU

Václav Hrbek

Bakalářská práce
2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2015/2016**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Václav Hrbek**
Osobní číslo:	**I13061**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Řízení procesů**
Název tématu:	**Jednoduché HMI vybavení pro monitorování reálného procesu**
Zadávající katedra:	**Katedra řízení procesů**

Zásady pro vypracování:

Bude vytvořeno zařízení na bázi univerzálního modulu s jednočipovým počítačem, umožňující monitorovat připojený reálný proces. Vstupy a výstupy procesu jsou napěťové signály v rozsahu 0-10V. Bude rovněž vytvořena aplikace v prostředí MS Windows, komunikující s modulem prostřednictvím sériového portu, která umožní nastavovat vstupy procesu v reálném čase a sledovat historii vstupů a výstupů.
Teoretická část bude obsahovat uvedení do problematiky, popis základních principů a algoritmů využitelných pro řešení problému.
Praktická část bude obsahovat:
1) popis hardware zvoleného pro implementaci rozhraní
2) vytvořený software ve zvoleném programovacím jazyce
3) ukázky práce s vytvořenou aplikací

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

HLAVA, J. Prostředky automatického řízení II: Analogové a číslicové regulátory, elektrické pohony, průmyslové komunikační systémy. Praha: Vydavatelství ČVUT, 2000.
CVEJN, J. Řízení procesů [online]. Pardubice: Univerzita Pardubice, FEI, 2012. Elektronický studijní materiál k předmětu Automatizace 1.

Vedoucí bakalářské práce: **doc. Ing. Jan Cvejn, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **19. listopadu 2015**
Termín odevzdání bakalářské práce: **13. května 2016**

Univerzita Pardubice Česká republika

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Daniel Honc, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2016

Prohlášení

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10.5.2017

Václav Hrbek

Poděkování

Rád bych poděkoval mému vedoucímu práce panu doc. Ing. Janu Cvejnovi Ph.D. za pomoc při návrhu aplikace.

V Pardubicích dne 10.5.2017

Václav Hrbek

ANOTACE

Práce se zabývá snímáním a následném řízení neelektrické veličiny. Výběrem vhodného hardwaru a stručném představení jeho alternativ. Dále bylo naprogramováno jednoduché uživatelské rozhraní.

KLÍČOVÁ SLOVA

Regulace, arduino, sériová komunikace, uživatelská aplikace.

TITLE

SIMPLE HMI EQUIPMENT FOR MONITORING OF THE REAL PROCESS

ANNOTATION

The work is oriented to scanning and subsequent regulation of non-electric value. By selection of appropriate hardware and a brief of presentation of alternatives. It was also programmed a simple human-machine interface.

KEYWORDS

Regulation, Arduino, Serial communication, User application.

OBSAH

SEZNAM ZKRATEK A ZNAČEK

HMI	human machine interface (Operátorské rozhraní)
C#	programovací jazyk
PC	persona computer (Osobní počítač)
PWM	pulse width modulation (Pulzně šířková modulace)
LED	light-emitting diode (Dioda emitující světlo)
PLC	programmable logic controller (Programovatelný logický automat)
MCU	microcontroller unit (Mikroprocesoravá jednotka)
GUI	graphical user interface (Uživatelské rozhraní)

SEZNAM SYMBOLŮ PROMĚNNÝCH VELIČIN A FUNKCÍ

R elektrický odpor, Ω

U elektrické napětí, V

SEZNAM ILUSTRACÍ

SEZNAM TABULEK

ÚVOD

K dnešním požadavkům na regulaci a monitorování běžných veličin, jako jsou například teplota, svítivost či vlhkost, narůstá i uživatelský požadavek na jejich jednoduché a přehledné ovládání.

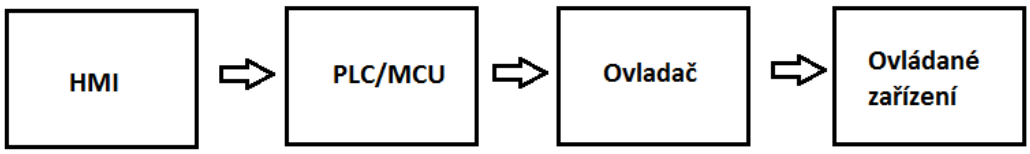
Cílem práce bylo vytvořit uživatelsky přívětivou aplikaci, která komunikuje s platformou Arduino po sériové lince. Zároveň by tato aplikace měla splňovat všechny požadavky na regulaci řízeného procesu pomocí HMI.

Práce je rozdělena na teoretickou a praktickou část. V teoretické části je popsána problematika HMI, představení platformy Arduino, sériová komunikace, regulace a představení vybraných funkcí jazyka C#. Praktická část obsahuje již konkrétní řešení elektrického zapojení, představení a sestavení řízeného procesu a podrobný popis sestavení HMI aplikace.

Výsledná aplikace umožňuje zobrazování a odečítání reálných hodnot. Dále umožňuje manipulaci s měřenými daty, kde využívá regulačních pochodů, ukládání, načítaní hodnot a ruční ovládání akčních členů.

1 HMI ROZHRANÍ

HMI (*Human Machine Interface*) je rozhraní mezi uživatelem a řízeným procesem. HMI technologie se nejčastěji vyskytují v průmyslové automatizaci kde regulují průmyslový proces a nesou informace o výsledcích měření. Za HMI zařízení lze považovat např. termostat v kanceláři, ovládací panel od CNC stroje nebo velké obrazovky v elektrárnách. HMI zajišťuje vizuální prezentaci řídícího systému a sběr dat. Je možné zvednou produktivitu několika řízených procesů jejich centralizací do jednoho HMI rozhraní.



Obr. 1.1 - Blokové schéma HMI

Dnešní době se nejvíce využívá HMI na dotykových panelech, kde se dá libovolně měnit mezi zobrazenými procesy. V praxi se nejvíce vyskytuje použití HMI rozhraní pro výrobní procesy jako centralizovaná řídící jednotka pro výrobní linky. Tato jednotka může být vybavena záznamem událostí, přepínači a různými ovládacími prvky tak že uživatel může mít přístup do systému kdykoliv a za libovolným účelem. Pro výrobní linky kam má být implementováno HMI rozhraní je nejprve třeba zajistit ovládání linky pomocí PLC nebo MCU. Toto PLC nebo MCU získává informace ze senzorů a následně je zpracovává pro HMI rozhraní, na jejímž základě může vyhodnocovat a rozhodovat o budoucím vývoji.

HMI vybavení může být použito jako:

- tlačítkové ovládání,
- zobrazovací panel,
- kombinací zobrazovacího panelu a ovládání.

Vývoj HMI se urychlil zejména v několika posledních letech díky rozšíření dotykových obrazovek. Před tím bylo HMI sadou několika desítek až stovek různých tlačítek a diod, které informovaly o řízeném procesu. Dotykový panel, na který se umístí vhodné kategorie s různými ovládacími prvky dokáže zpřehlednit a zrychlit výrobní proces.

Kombinací zobrazování vstupních hodnot a následném nastavení ovládacích veličin, lze získat optimální regulační a informační zařízení.

1.1 POUŽITÍ HMI ROZHRANÍ

HMI se používá v celé řadě různých průmyslových odvětvích, od prodejních automatů, přes celý potravinářský průmysl, farmaceutický průmysl, atd.. HMI společně s PLC nebo MCU jsou obvykle páteří výrobní linky v těchto odvětvích. Integrace HMI do výrobního procesu výrazně zrychlila prováděné operace na výrobní lince i mimo ni. HMI umožňuje dispečerům řízení a sběr dat v celém systému, takže dispečerovi umožňuje provádět změny parametrů řízeného systému dle svého uvážení. HMI nabízí lepší kontrolu zásob a tím zvýšení efektivity výrobního procesu.

1.2 HISTORIE HMI

Produkty HMI vznikly z potřeby, aby strojní zařízení bylo jednodušší a přehlednější na ovládání. Předchůdce HMI byl dávkový soubor. Ten je však neinteraktivní, funguje tak, že uživatel zadá sadu příkazů, které se mají zpracovat, ale během zpracování nemůže uživatel do procesu zasahovat.

Dalším předchůdcem plnohodnotného HMI bylo zadávání do příkazového řádku pomocí klávesnice. S příchodem příkazového řádku přišla interakce v průběhu procesu. To však bylo pro běžné uživatele stále příliš složité.

S nástupem GUI bylo možné rozšířit HMI mezi běžné uživatele. GUI umožňuje uživateli komunikovat s procesem více způsoby, než jen pomocí psaní příkazů. GUI samotné pochází z nutnosti obsluhovat stroje mnohem efektivněji.

1.3 HARDWAROVÉ POŽADAVKY NA HMI

Při konstrukci nebo výběru HMI vybavení a jeho následné integraci je vhodné brát ohled na to v jakých podmínkách a s jakými nároky na hardware bude použito.

1.3.1 Výběr dle okolního prostředí

Okolní prostředí má zásadní vliv na konstrukci HMI zařízení. Pokud bude HMI vybavení v prašném prostředí, vyplatí se vybrat součástky na konstrukci, které jsou odolné vůči prachu. Ve vlhkém prostředí je třeba zajistit vodotěsný obal HMI vybavení. Pokud bude

HMI použito ve vysokých teplotách, je nutné zajistit dostatečné chlazení. Nadměrné vibrace či hluk je potřeba také ošetřit pomocí speciálních součástek.

1.3.2 Výběr dle požadavků na hardware

HMI vybavení je nutné volit s ohledem na kladené požadavky. Pokud je nutné zpracovávat několik tisíc hodnot ze senzorů a následně regulovat hodnoty veličin, je vhodné mít rychlejší hardwarovou výbavu. K HMI zařízení může být připojena externí paměť, kam se boudou ukládat naměřené hodnoty.

1.4 SOFTWAROVÉ VYBAVENÍ HMI

Při výběru softwarového vybavení HMI je na výběr z několika možností. Uzavřený software je software dodávaný výrobcem ke konkrétnímu hardwarovému vybavení, mezi jehož výhody patří, snadné programování, nastavení a odladěný od chyb. Ve velké míře má v sobě už předpřipravené různé nástroje pro ovládání systému. Do uzavřeného softwaru není možné zasahovat a provádět změny, které nejsou umožněny vydavatelem softwaru.

Opakem uzavřeného softwaru je otevřený software. To je programové vybavení, jenž vzniká, nejčastěji na základě určitého zadání, pomocí různých programovacích jazyků, např. C a C++. Tento software je nesmírně modulární a téměř neomezuje HMI v požadavcích na řízení a regulaci procesu. Tvorba tohoto softwaru je doporučena jen zkušeným programátorům a vývojářům.

2 VYUŽITÉ NÁSTROJE A TECHNOLOGIE

2.1 PROGRAMOVACÍ JAZYK C#

Programovací jazyk C# byl vyvinut a je vyvíjen firmou Microsoft. Byl uveden s platformou *.NET Framework.* Jedná se o vysokoúrovňový programovací jazyk, který vychází z programovacích jazyků *C* a *C++*. V mnoha ohledech čerpá z programovacího jazyka *Java.*

Jazyk C# je objektově orientovaný, nejčastěji se využívá při tvorbě webových aplikací a stránek, webových služeb, tvorbě databázových programů, formulářových aplikací ve

Windows apod. Programovací jazyk C# dále obsahuje podporu komponentového programování. Jazyk je navržen tak, že se využívá pro psaní aplikací u zařízení s operačním systémem, ale také pro vestavěná zařízení. C# je integrován do prostředí Visual Studio.NET.

2.1.1 Historie C#

V roce 2002 představila společnost Microsoft první verzi jazyka C# společně s .NET Frameworkem. Vycházel tehdy z jazyka C++, ze kterého si bral hlavně podporu objektově orientovaného programování a zkušenost z jazyka Java s jeho aktualizacemi.

Druhá verze byla vydaná na konci roku 2005. Zde byly integrovány anonymní metody, statické a částečné třídy.

O dva roky později, tedy v roce 2007, vyšel společně s .NET Frameworkem 3.5 a tehdy nejnovějším Visual Studiem 2008 C# 3.0. Změny, kterých se C# 3.0 dočkal, byly opravdu velké. Pokud byly na daném zařízení určité knihovny, bylo možné spustit aplikace z jazyka C#, navzdory tomu, že zařízení obsahovalo pouze druhý Framework. Dále zde byl implementován integrovaný dotazovací jazyk *LINQ (Language Integrated Query)*.

LINQ významně usnadňuje tvorbu, třídění a vyhledávání dat. Tento nejnovější způsob dotazování na jakákoliv data zjednodušil jejich práci s nimi, bez ohledu na zdroj dat. LINQ to Objects umožňuje dotazování na celé kolekce objektů. Pro práci s databází SQL je nadmíru užitečné dotazování LINQ to SQL.

V roce 2010 vyšla verze jazyka C# 4.0. S touto verzí nastoupilo dynamické programování a nová práce s Frameworky. Mezi novinky přibyly dynamicky typované objekty, volitelné parametry, parametry určené jménem, vylepšená COM interoperabilita a kovariance a kontravariance.

Další verzí byl C# 5.0. Tato verze byla vydána v roce 2012, společně s .NET Frameworkem 4.5. a vývojovým prostředím Visual Studio 2012. Hlavní novinkou byla zejména možnost asynchronního programování.

Nejnovější je verze C# 6.0. Verze vyšla v roce 2015. Spolu s touto verzí bylo vydáno i nové Visual Studio 2015 a .NET Framework 4.6. Tato verze toho moc nového nepřidává, ale spíše se zaměřuje na čitelnost a přehlednost kódu.

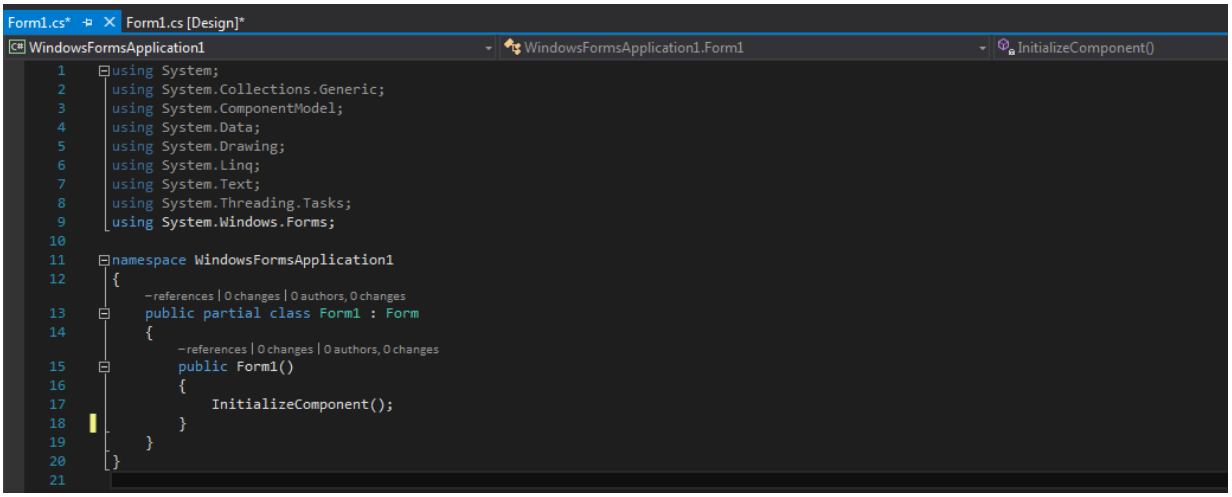
2.1.2 Vlastnosti jazyka C#

Mezi hlavní charakteristické rysy jazyka C# patří, že jako programovací jazyk je čistě objektově orientovaný, má v sobě implementovanou podporu komponentového programování, dokáže ošetřit hranice polí a detekovat neinicializované proměnné. C# obsahuje pouze jednoduchou dědičnost s možností násobné implementace, jako je tomu u programovacího jazyka Java. Kromě členských dat a metod přidává vlastnosti a události. V jazyce C bylo dle potřeby nutné alokovat dostatek paměti pro program a práci s ním. V jazyce C# je automatická správa paměti. O alokaci zdrojů a jejich následné uvolňování se stará *garbage collector*. Pomocí výjimek lze ošetřit zpracování chyb. C# zároveň zajišťuje typovou bezpečnost a zpětnou kompatibilitu se staršími verzemi, a to jak na binární, tak na zdrojové úrovni.

Jazyk C# rozlišuje velká a malá písmena. Jeho překladače jsou *case sensitive.*

2.1.3 Konvence jazyka C#

Jako v mnoha programovacích jazycích, tak i programovací jazyk C# má svojí typickou konvenci.



Obr. 2.1 – Program generovaný při vytvoření formulářové aplikace

Velkým písmenem se píší jména tříd, rozhraní, balíků a většiny dalších položek. Malým písmenem se píší privátní a chráněné atributy, lokální proměnné a parametry. Více o konvencích tohoto jazyka je k nalezení v dokumentaci *Naming Guidelines.*

Obr. 2.1 znázorňuje jednoduchou kostru programu, která je generována programem Visual Studio 2015, po vytvoření nové formulářové aplikace.

2.1.4 DLL

DLL (*Dinamic-link Library*) je soubor, který pracuje jako knihovna funkcí. Tato knihovna může být sdílena pro víc programů. Pomocí DDL lze volat funkci, jež není součástí programu, tudíž není součástí kódu, ale je umístěna v DDL. Může obsahovat více funkcí. Tyto funkce se kompilují odděleně od procesů. Pomocí dynamické knihovny lze sdílet data a program. Dynamickému propojení náleží pouze informace, které jsou nutné k nalezení spustitelného kódu pro funkci knihovny DLL. Knihovna DLL podporuje vícejazyčné programy, výrazně šetří paměť, a tak spustitelný program zabírá méně místa na disku.

2.1.5 .NET Framework

Microsoftem vyvinutý framework .NET, je kolekce několika rozhraní pro programování aplikací a sdílených knihoven, které mohou být vývojářem využity při programování aplikací a nemusí je psát od začátku. Platforma je určena nejen vývoj aplikace na operační systém Windows, ale i pro webové aplikace a aplikace pro mobilní zařízení. Knihovny se sdíleným kódem se nazývají *Framework Class Library*. Dále framework obsahuje samostatné běhové prostředí (*Comon Language Runtime*), které se stará o kompilaci a běh aplikace.

Kompilace aplikace v .NET je odlišná než kompilace v C++. Kompilátor ze zdrojových kódů neudělá strojový kód pro dané zařízení, ale přeloží ho do *Common Intermediate Language* (*CLI*). Výhodou tohoto kódu je, že je nezávislé na platformě a dá se spustit všude, kde je instalován .NET Framework. Další odlišností je že .NET využívá při kompilaci *Just-in-time* překladač, který překládá jen tu část kódu potřebnou při volání aplikace.

2.1.6 Prvky jazyka C#

Základní prvky jazyka C# jsou podobné jako v jiných programovacích jazycích. Zde jsou uvedeny jen ty nejzákladnější, které se nejčastěji používají a jsou využity v praktické části práce.

Jedním ze základních prvků jazyka je třída, což je propojení metod, událostí a proměnných. Implementace tříd v jazyce Java se neliší od implementace tříd v jazyce C#. Uživatel si může vytvořit vlastní typ třídy, díky které je možné definovat její chování a data. Třídou lze vytvořit instanci a objekty, které jsou přiřazeny k proměnným. Pokud lze v paměti nalézt odkaz na objekt nebo instanci, tak zůstává zapsaný v paměti. Jestli odkaz neexistuje, *CLR* uvolní paměť. Třídy deklarované jako statické v paměti existují jen jednou.

Dalším základním prvkem jazyka C# je pole. Pod tímto pojmem si lze představit množinu stejného datového typu. Tato množina tvoří celek. S polem se zachází jako s celkem, a na jednotlivé prvky (jednotlivé proměnné v poli) se odkazuje pomocí jejich indexů v poli. Čtení pomocí indexu v poli, ze kterého se vrátí proměnná, umožní práci s prvkem jako s obyčejnou proměnnou. Prvnímu prvku v poli náleží index *0*. Celková délka pole má označení *N*. Pokud tedy bude pole délky *N*, tak poslední prvek v poli bude mít index *N-1*. C# definuje dva druhy pole. Pole vícerozměrné a pole jednorozměrné. Instance pole se vytváří pomocí statické metody *System.Array.CreateInstance()*. Jednorozměrná pole se v jazyce C# deklarují následujícím způsobem:

- *typ_pole[] název_pole = new typ [velikost_pole];*

Dvourozměrné pole se deklaruje následovně:

- *typ_pole[] název_pole = new typ [velikost_pole_x, velikos_pole_y];*

Jazyk C# neumožnuje vícenásobnou dědičnost. Aby bylo možné přiřadit třídám různé charakteristiky nebo schopnosti nezávisle na hierarchii tříd, je nutné použít rozhraní. Deklarace rozhraní se provede klíčovým slovem *interface*. Po tomto slovu následuje jméno rozhraní. Samotné tělo rozhraní se pak píše do složených závorek, ve kterých se nachází výčet metod, vlastnosti a události. Členy rozhraní jsou vždy veřejné (*public*). V jazyce C# lze implementovat třídám více rozhraní. Rozhraní samotné lze rozšířit užitím dědičnosti, a díky tomu slučovat více rozhraní do jednoho a přidávat další metody.

Jazyk C# má téměř stejnou množinu operátorů jako jiné programovací jazyky, například Java. Tab. 2.1 uvádí výčet nedůležitějších operátorů, které jsou seřazené dle priority použití.

Tab. 2.1 – Operátory

Primární	new, typeof, sizeof, checked, unchecked, (x) x.y f(x) a[x] x++ x--
Unární	+ - ! ~ ++x, --x (T)x
Multiplikativní	* / %
Aditivní	+, -
Bitové posuny	<< >>
Relační	< > <= >= is as
Rovnost	== !=
Logické AND (bitové)	&
Logické XOR (bitové)	^
Logické OR (bitové)	|
AND	&&
OR	||
Podmíněný výraz	? :
Přiřazení	= *= /= %= +- -= <<= >>= &= ^= |=

Výjimky jsou odvozeny ze třídy *Object*. Základní třída výjimek, která je z této třídy odvozena, je třída *Exception*. Z této třídy dědí ostatní výjimky.

Výjimky nachází uplatnění v případě neočekávaného běhu programu, např. dělení nulou nebo snaha zapsat prvek mimo hranice pole. Lze je chápat jako objekty, které nesou informace o vzniklém problému. Výjimky je možné vyvolat v prostředí .Net Framework, nebo se dají vyvolat uživatelem pomocí příkazu *throw*.

Pro odchyt výjimek se používá následující struktura bloku *try*.

```
try
{
      //Příkazy pro kontrolu
}
catch (System.NullPointException e)
{
      //Odchycení výjimky NullPointException, a reakce na ni
}
catch (System.Exception e)
{
      //Odchycení výjimky Exception, a reakce na ni
}
```

2.2 METRO FRAMEWORK

MetroFramework je nový design od společnosti Microsoft. Tento nový koncept změnil logické rozložení a práci s operačním systémem Windows 8 a jeho aplikacemi. Základní principy a vzhled byly ve velkém převzaty z informačních systémů v transportních stanicích. Při vzniku byl kladen důraz na čitelnost, zřetelnost a vynechání nepřehledných částí. Základní myšlenkou bylo rychle, bezpečně a bez zbytečného rozptylování nalézt všechny uživatelem požadované informace a zanechat příjemný pocit po shlédnutí dané aplikace.

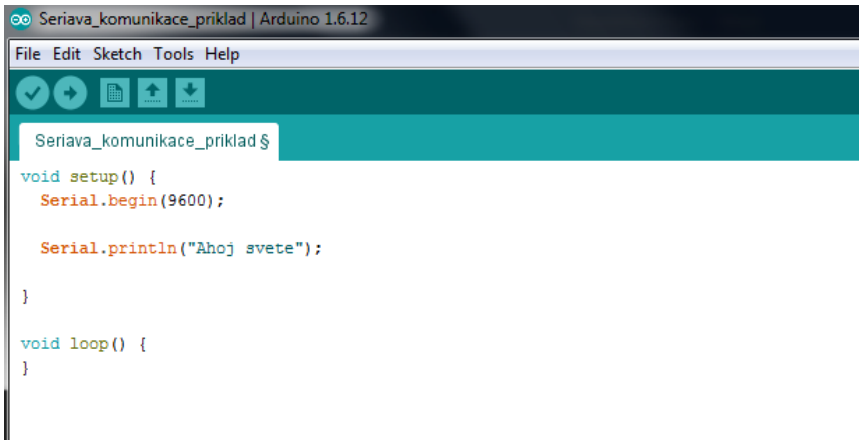
Hlavním principem je tedy zaměření se na koncového uživatele. Uživateli, který používá elektronické zařízení, je jedno, jestli chce přistupovat k datům nebo vyřešit rychle své úlohy, tyto úkony musí plnit pohodlně a být minimálně rozptylován. Uživatelské rozhraní ztrácí svou hodnotu, pokud uživatel musí složitě hledat cestu k požadované funkci. Společnost Microsoft tedy odstranila pozadí, většinu dekoračních prvků, díky kterým bylo obtížné najít hledanou věc a nechala jen důležité věci. Tyto principy by měly ulehčit navigaci koncového uživatele na hledanou informaci.

2.3 MIKROPOČÍTAČ ATMEGA328

Mikropočítač ATmega328 obsahuje řadu užitečných integrovaných obvodů, které pomáhají s obsluhou vstupů a výstupů. ATmega328 používá 8-bitové jádro. Dále má k dispozici 1 kB EEPROM paměti, 32 pracovních registrů, vnitřní a vnější přerušení, čítač/časovač, sériové rozhraní USART, dvou drátovou sériovou sběrnici, SPI sériový port, 10 bitový A/D převodník a programovatelný *watchdog* s vnitřním oscilátorem. Napájecí napětí ATmega328 je 5 V. *Microcontroller* je vybaven 28 vstupně-výstupními piny.

2.3.1 Sériová komunikace

Arduino desky komunikují s PC a jinými zařízeními pomocí sériové komunikace. Všechny Arduino desky mají alespoň jeden sériový port. Tento port je většinou USB port. Čipy, kterými jsou desky osazené, používají komunikační rozhraní UART nebo USART a pracují s TTL logickými hodnotami napětí, které mohou nabývat hodnot 5 V nebo 3,3 V. To, jaké napětí se používá, záleží na typu desky. Poté, co je vyslán signál od čipu, je převeden na napěťovou úroveň, jenž odpovídá sériovému rozhraní RS232. Toto napětí je +12 V a -12 V.

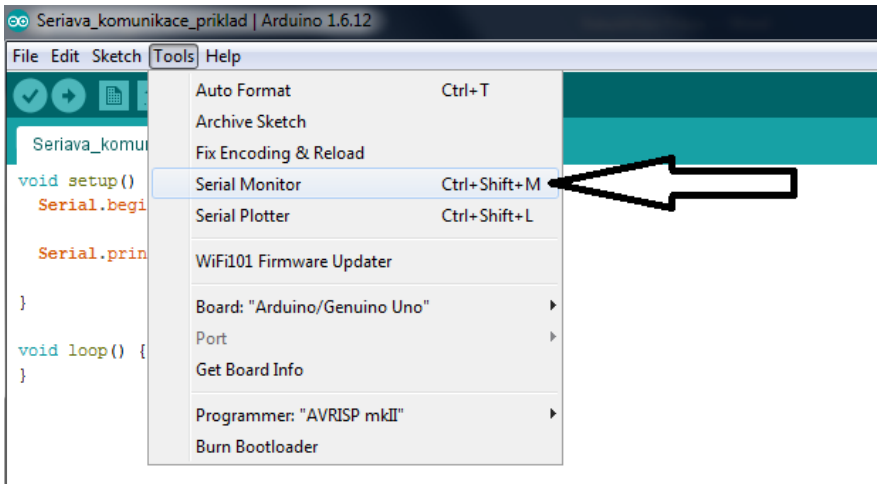


Obr. 2.2 – Příklad použití sériové komunikace

V Arduino IDE se využívá knihovna *Serial*(), která obstarává veškerou komunikaci mezi PC a Arduinem.

Na obr. 2.2, je vidět příklad použití sériové komunikace. Příkaz v nastavení *Serial.begin(9600);* zahajuje sériovou komunikaci. Číslo 9600 v závorkách je rychlost komunikace (anglicky nazýváno *baud rate*). Vyjadřuje se v jednotkách *bps* neboli *bits-per-second*.

Příkaz *Serial.println("Ahoj svete")* pošle přes sériové rozhraní celý řádek s textem "Ahoj svete".



Obr. 2.3 – Seriál Monitor 1

Arduino IDE má nástroj na sledování sériové komunikace. Je k nalezení pod názvem *Serial Monitor* na pracovní liště nebo spustitelný klávesovou zkratkou Ctrl+Shift+M. Kde spustit *Seriál Monitor* znázorňuje obr .2.3.

2.3.2 Čítač/Časovač

Přerušení pomocí časovače se používá v případech, kdy je potřeba vykonat operaci nezávisle na běhu hlavního programu. Při práci s přerušením se nastavují registry přímo v mikročipu ATmega328, takže není využita žádná speciální knihovna Arduino IDE. V mém programu používám přerušení, aby došlo k odeslání a přijmutí hodnoty ze sériové komunikace. K tomu je vyvoláno přerušení, při kterém čítač dosáhne shodné hodnoty s porovnávaným registrem. Deska Arduino Uno má 3 časovače, Timer0, Timer1 a Timer2. Každý z nich má čítač, který se inkrementuje při přetečení časovače. Poté je časovač vynulován a připraven k dalšímu měření. Výběrem porovnávané hodnoty lze určit rychlost, jakou se bude čítač inkrementovat a tím je možné nastavit frekvenci přerušení.

Je mnoho parametrů, které je potřeba zohlednit při nastavování čítače/časovače. První z nich je rychlost, při které časovač inkrementuje čítač. Desky Arduino jsou vybaveny externím krystalem pro taktování procesoru. Díky tomuto krystalu jsou mikročipy nastaveny na frekvenci 16 MHz. To znamená, že pokud časovač běží na maximální frekvenci, tak se čítač inkrementuje přibližně každých 63 ns. V čipu ATmega328 jsou dva 8 bitové časovače,

Timer0 a Timer2 a ty nabývají maximální hodnoty až 255. Timer1 je 16 bitový a jeho maximální hodnota je 6553. Jakmile čítač dosáhne své maximální hodnoty, tak se vynuluje a je možné do něj opět zapisovat. Například, je-li pracovní frekvence 16 MHz, tak 8 bitový časovač přeteče každých 16µs a 16 bitový každé 4ms. Z výše uvedeného textu vyplývá, že není možné přímo kontrolovat rychlost inkrementace čítače, ale je nutné použít předděličku.

Předdělička snižuje pracovní frekvenci mikročipu a nabývá hodnot 1, 8, 64 a 1024. Může snížit pracovní kmitočet až na hodnotu 15625 Hz a nastavuje se v porovnávacím registru TCCR1B, a to bity CS11, CS12 a CS10.

2.4 ARDUINO

Arduino se začalo vyvíjet v roce 2005 v Italském městě Ivera. Cílem bylo vytvoření jednoduché a levné platformy pro studenty. Arduino se stalo mezi studenty oblíbené, tak se tvůrci rozhodli pro rozšíření mezi celosvětovou veřejnost. Arduino je *Open Source* projekt. To znamená, že tvůrci sdílejí všechna schémata a návody se širokou veřejností. Díky tomu vznikl nespočet neoficiálních typů desek, takzvaných klonů (Voda, 2015).

Obr 2.4 – Oficiální logo Arduina

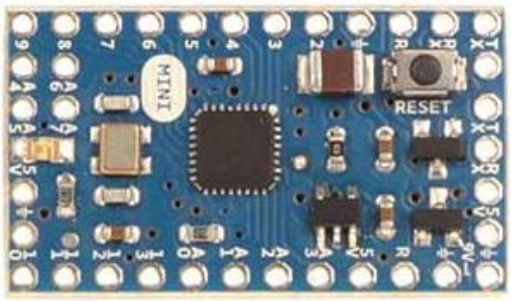
Pomocí Arduina lze získávat údaje od různých snímačů a senzorů a na základě těchto údajů ovládat výstupy. Zvláštními typy desek jsou Arduino *Shieldy*. *Shieldy* jsou rozšiřovací deskou pro základní desku Arduina. Například se používají pro ovládaní motorů nebo pro komunikaci pomocí Ethernet portu (Arduino.cz, 2014 - 2017).

Mezi výhody Arduino platformy patří nízká cena, obrovské množství kompatibilního hardwaru a multiplatformnost. Nějvětší výhodou je ovšem obrovská komunitní základna nadšenců, díky které lze nalézt řešení na jakýkoliv problém (Arduino.cz, 2014 - 2017).

Arduino obsahuje procesory od firmy Atmel. Kromě procesoru obsahují desky celou řadu dalších komponent. Jednotlivé verze desek se mohou lišit v rozložení součástek nebo designu desky. Desky Arduino mají typické grafické zpracování s modrou barvou. Mimo hlavního čipu desky obsahují ještě převodník s čipem. Některé desky však tento převodník nemají kvůli úspoře místa (Voda, 2015).

2.4.1 Arduino desky

Nejmenší oficiální verze Arduina je Arduino Mini. Arduino Mini bylo navrženo tak, aby co nejvíce šetřilo místem. Své využití nachází v chytrých vypínačích a jiných malých zařízeních. Používá procesor ATmega328 s taktem 16 MHz. Kvůli úspoře místa byl odstraněn USB port. Aby bylo možné Arduino Mini programovat, je nutné použít externí sériový převodník. Výkon není jeho velikostí nijak ovlivněn a může konkurovat větším deskám (Voda, 2015).



Obr. 2.5 – Arduino Mini (Voda, 2015)

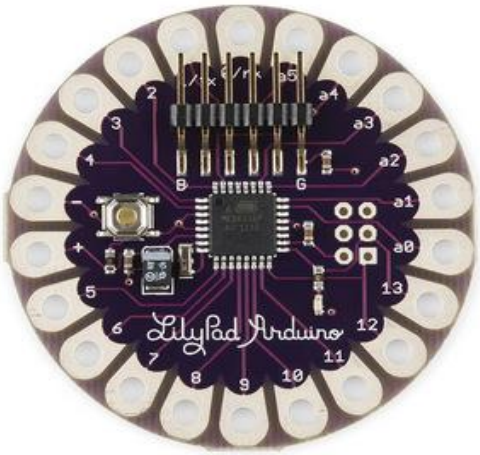
Arduino Nano je o něco větší než Arduino Mini. Hlavním rozdílem je možnost programování Arduina Nano pomocí integrovaného USB portu. Není tedy nutné vlastnit společně s deskou ještě externí USB programátor. Ostatní výbavou se Arduino Nano a Arduino Mini výrazně neliší (Voda, 2015).

Obr. 2.6 – Arduino Nano (Voda, 2015)

Arduino Micro je jedna z desek, která již obsahuje čip s převodníkem. Tento čip je ATmega32u2. Odpadá nutnost externího USB převodníku. Deska se tváří pro počítač jako externí zařízení. Použitím této desky je možné vytvořit vlastní klávesnici nebo myš. Umožňuje posílat příkazy jako stisk klávesy nebo posunutí myši. Deska je přizpůsobena pro zasunutí do nepájivého kontaktního pole. Arduino Micro se funkčně shoduje s deskou Arduino Leonardo (Voda, 2015).

Obr. 2.7 – Arduino Micro (Voda, 2015)

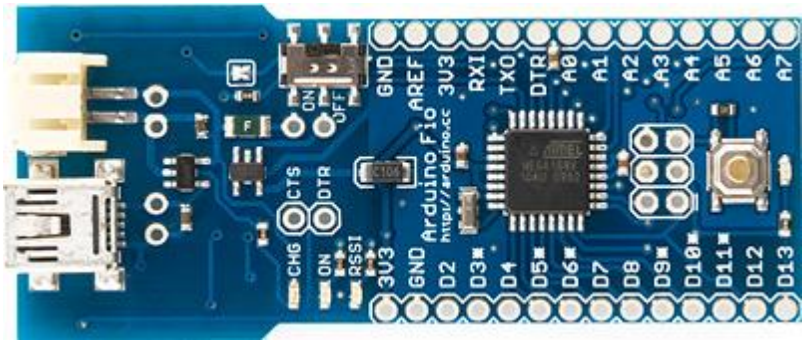
Deska nazvaná LyliPad je na první pohled odlišná než předešlé desky. LyliPad Arduino je přizpůsobeno pro připevnění na textil pomocí vodivých nití. LyliPad Arduino



Obr. 2.8 – LyliPad Arduino (Voda, 2015)

existuje ve verzích s čipem ATmega32u4 nebo ATmega328. Pokud je potřeba vytvořit oděv s elektronickými reflexními prvky, je tato deska nejlepší možnou volbou (Voda, 2015).

Deska Arduino Fio byla vytvořena pro připojení speciálních bezdrátových modulů. Takzvané *XBee* moduly. Arduino Fio je vybaveno čipem ATmega328P s frekvencí 8MHz. Zajímavostí je napájecí napětí této desky, které je 3,3 V. Toto napětí bylo zvoleno kvůli kompatibilitě mezi deskou a moduly (Voda, 2015).



Obr. 2.9 – Arduino Fio (Voda, 2015)

V dnešní době je Arduino Uno nejrozšířenější deska z nabízených desek. Tato deska obsahuje klasický USB port. Srdcem desky je čip ATmega328. Arduino Uno je přímým nástupcem hlavní vývojové linie, jejímž prvním členem bylo Arduino se sériovým portem USB. Tato linie se táhne přes Arduino Extreme, NG, Diecimila a Duernilanove až k dnešnímu

Obr. 2.10 – Arduino Uno (Voda, 2015)

Arduino Uno. Hlavní linie se rozčlenila na více desek, a to na Arduino Ethernet, Arduino Bluetooth a Arduino Pro. Jak lze poznat z názvu, u Arduina Ethernet byl nahrazen USB port Ethernet portem. Arduino Bluetooth je přednostně určeno pro bezdrátovou komunikaci. Arduino Pro je ochuzenou verzí Arduina Uno. Chybí mu USB převodník. K jeho programování je tedy potřeba externí převodník. Používá se hlavně pro pevné zabudování do nějakého projektu (Voda, 2015).

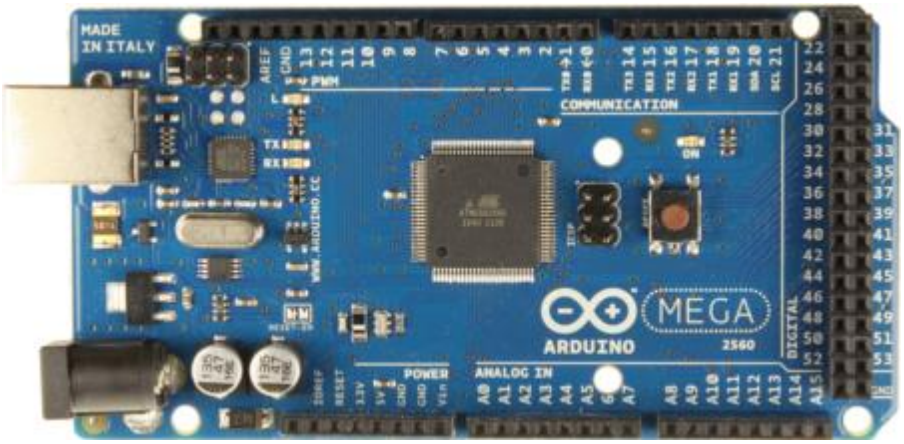
Speciální odnoží Arduino Uno je Arduino Leonardo, které se liší v použitém čipu ATmega32u4. Tento čip se používá u Arduino Micro (Voda, 2015).

Deska Arduino Yún je naprosto unikátním modelem Arduino platformy. Arduino Yún obsahuje 2 různé čipy. Těmi čipy jsou ATmega32u4 a AtherosAR9331. Čip AtherosAR9331 je schopný rozeběhnout odlehčenou verzi Linuxu Linio. Kvůli zajištění komunikace mezi oběma čipy obsahuje Arduino Yún softwarový *brige*. Deska Arduino Yún designově navazuje na desku Arduino Uno. Součástí desky je microUSB pro programování ATmega32u4 a

Obr. 2.11 – Arduino Yún (Voda, 2015)

normální USB pro programování AR9331. Dále tam lze najít Ethernet port pro připojení desky do sítě. Arduino Yún je možné použít pro snímání naměřených hodnot a jejich následné odeslání na webový server (Voda, 2015).

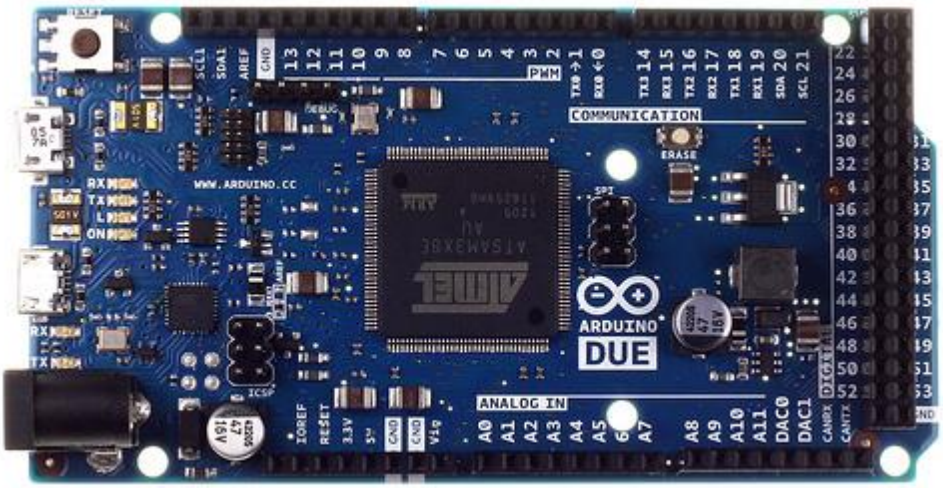
Arduino Mega2560 je zvětšená verze Arduina Uno. Deska je prodloužená a vybavena výkonnějším čipem. Prodloužení desky také znamená, že se na ní vejde více pinů. Deska



Obr. 2.12 – Arduino Mega2560 (Voda, 2015)

obsahuje 54 digitálních a 16 analogových pinů. Použití této desky je vhodné zejména v případech, kdy je potřeba větší výpočetní výkon. Zajímavá je deska Arduino Mega ADK, která je odvozena od této desky, ale má navíc jeden USB port. Tak je možné na desku připojit externí zařízení, např. Android (Voda, 2015).

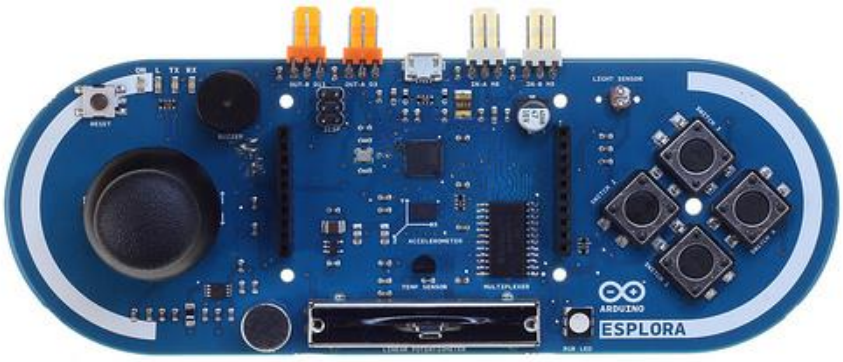
Nástupcem Arduino Mega je velice výkonné Arduino Due. Hlavní předností Arduino Due je čip AtmelSAM3X8E. Tento čip má oproti běžně používaným 8–bitovým čipům 32–bitové jádro. Tím se také zvýšila taktovací frekvence z 16MHz na 84MHz. Deska je dále vybavena dvěma microUSB konektory. Z jednoho je možné programování tohoto výkonného



Obr. 2.13 – Arduino Due (Voda, 2015)

čipu a druhý může být použit pro připojení externích zařízení jako je klávesnice, myš, zařízení s Androidem a jiné. Deska disponuje 54 digitálními piny a 12 analogovými piny (Voda, 2015).

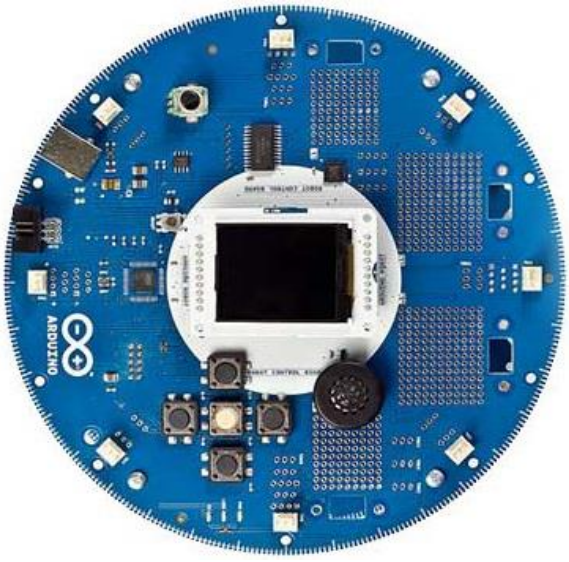
Arduino Esplora vypadá designově jako herní ovladač. Na desce zaujme joystick a čtyři tlačítka. Deska dále obsahuje posuvný potenciometr, piezo bzučák, teploměr, tříosý



Obr. 2.14 – Arduino Esplora (Voda, 2015)

akcelerometr a piny pro připojení LCD displeje. Díky tomuto vybavení je možné zařadit desku do hybridní kategorie. Srdcem desky je čip ATmega32u4. Z této desky je možné pohodlně vytvořit herní ovladač pro ovládání herní konzole (Voda, 2015).

Arduino Robot není jen deska, ale je to celý set se senzory a akčními členy. Tento set je vhodný pro stavbu a vytvoření vlastního chytrého robota. Robot obsahuje dvě samostatné



Obr. 2.15 – Arduino Robot (Voda, 2015) 1

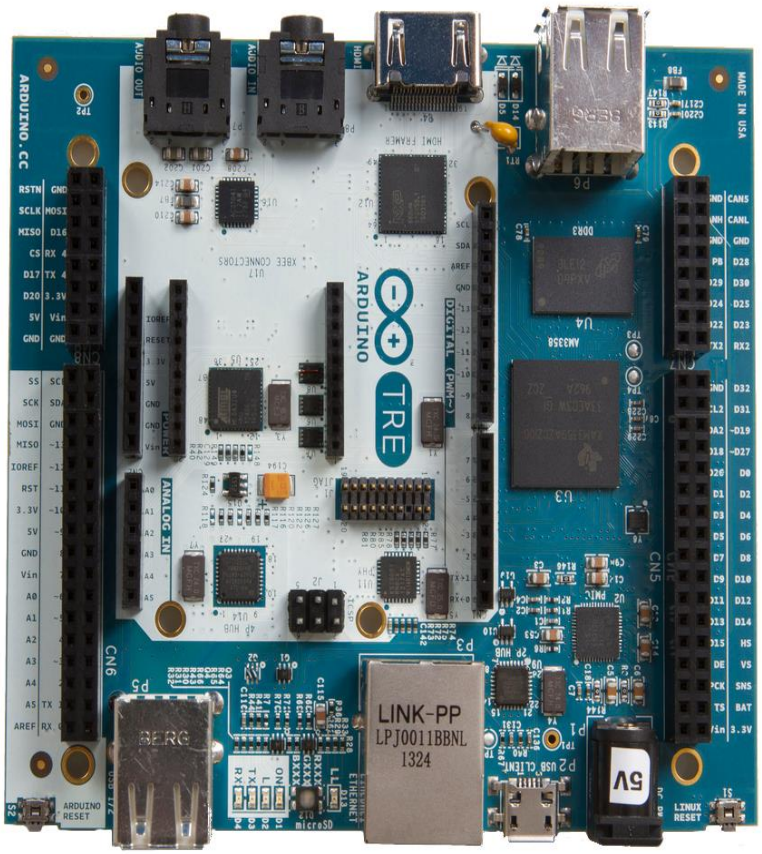
desky. Každá je obsazena čipem ATmega32u4 a je možné je samostatně programovat. Jedna

deska robota se stará o řízení motorů. Druhá deska rozhoduje o řízení robota a vyhodnocuje data ze senzorů (Voda, 2015).

Arduino Inte Galileo vzniklo ze spolupráce společností Intel a Arduino. Deska je unikátní tím, že obsahuje čip Intel QuarkSoCX1000. Tento čip má 32–bitové jádro a pracuje na frekvenci 400 MHz. Deska je osazena množstvím příslušenství. Obsahuje dvě USB, microSD slot a Ethernet port. Nalezneme zde i mini–PCI Express slot, který lze využít pro připojení externích karet (Voda, 2015)

Obr. 2.16 – Arduino Intel Galileo (Voda, 2015)

Arduino Tre je deska, která je v současné době testována. Je to nejvýkonnější deska z rodiny Arduino desek, je podobná Arduinu Yún, jelikož obsahuje dva procesory. Stejně, jako je tomu u Arduino Yún, je zde jeden čip použit pro jádro Arduina a jeden pro jádro Linuxu. Arduino Tre obsahuje HDMI port, dva audio konektory a jeden USB port pro programování. Dále zde nalezneme čtyři porty pro připojení externích zařízení (Lehrbaum, 2013)



Obr. 2.17 – Arduino Tre (Lehrbaum, 2013)

2.5 ARDUINO IDE

Arduino IDE je integrované vývojové prostředí pro programování Arduino platformy a jeho klonů. Je psané v jazyce Java, a tak je to multiplatformní prostředí, které lze nainstalovat na různé typy operačních systémů, např. Mac OS, Linux a Windows. Samotné programování probíhá v jazyce C a C++. Tento otevřený software vznikl z výukového prostředí Processing, které vzniklo pro přiblížení programování i „neprogramátorům“. Prostředí Processing bylo následně upraveno a byly mu přidány některé užitečné funkce. Za zmínku stojí podpora knihovny Wiring, která je hromadně rozšířená a komplexní a je oblíbená z důvodu dostupnosti mnoha návodů. Knihovna Wiring je někdy označována jako samostatný programovací jazyk (Voda, 2015).

3 PRAKTICKÁ ČÁST

Cílem praktické části této práce je vytvořit HMI aplikaci, která bude obsahovat všechny požadované funkce běžného průmyslového HMI. Práci této aplikace pak ověřit na zapojení řízeného procesu.

Aplikace by měla obsahovat zobrazovací prvek, ve kterém by bylo možné vidět a sledovat průběh procesu. Interakce s tímto zobrazovacím prvkem by měla být realizována pomocí počítačové myši, a to tak, že najetím myši na měřenou hodnotu se zobrazí informace o této hodnotě. Dále je pak nutné sledovat historii řízeného procesu pomocí zobrazovacího prvku v libovolném časovém úseku. Historie měření by se měla dát ukládat do textového souboru a následně tento textový soubor načíst a zobrazit v grafu. V několika textových polích, se budou zobrazovat hodnoty, jak aktuálně naměřené, tak i maximální a minimální za celou dobu měření. V regulaci je žádoucí nastavování žádané veličiny a pásma necitlivosti pomocí textových polí. Neměla by chybět možnost výběru mezi dvěma režimy, automatickou regulací, nebo ruční regulace akčního členu.

3.1 ŘÍZENÝ PROCES

Praktické zapojení a aplikace vytvořená pro tuto práci má za úkol měřit, regulovat a simulovat intenzitu osvětlení v místnosti. Na vybrané zapojení by se dalo aplikovat velké množství senzorů měřících různé veličiny, které se v reálném průmyslu běžně vyskytují, a pro své elektrické zapojení mají podobnou napěťovou logiku jako zde vybraný optoelektronický senzor. To znamená, že pokud by byl místo fotorezistoru zapojen termistor, tak by se dala se stejným zapojením měřit teplota.

K měření osvětlení je použit fotorezistor, polovodičová součástka, jejíž odpor se snižuje při větší intenzitě osvětlení. K simulaci osvětlení byla použita jedna LED dioda, která dokáže v tomto procesu plnohodnotně nahradit externí zdroj osvětlení. Fotorezistor snímá světlo z LED diody a z místnosti, kde se vyskytuje přirozené denní osvětlení. Poněvadž nebylo použito žádné vedlejší stínění, tyto dva zdroje světla dávají dohromady výslednou intenzitu osvětlení, kterou fotorezistor snímá.

Z předchozího odstavce vyplývá, že regulovatelný zdroj osvětlení je jen jeden. Aby bylo možné simulovat stmívání denního světla a nebylo nutné fyzicky ztmavovat místnost, což je velice náročné na technické požadavky místnosti, ve které se měření provádí, byla

přidána metoda, která simuluje zatemňovací prvek. Metoda uměle snižuje hodnotu napětí kterou dodá optočlen.

Řízený proces snímání a regulace světla, je využitelný v mnoha případech. Prvním ukázkovým případem je jeho využití v moderních domech, kde je kladen důraz na co největší snížení spotřeby elektrické energie. Další využití měření osvětlení bývá při chemických procesech, které jsou náchylné na míru osvětlení. Ve velkých kancelářích, ve kterých je pohromadě několik pracujících zaměstnanců, lze snížit jejich únavu v průběhu dne pomocí regulace osvětlení.

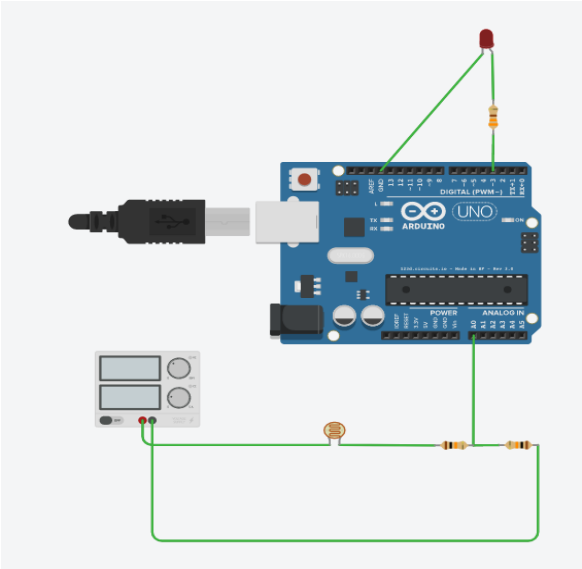
3.1.1 Elektrické zapojení řízeného procesu

Aby bylo možné ověřit funkčnost programu a výsledné aplikace, byly zapojeny fyzické součástky na nepájivém poli. Podrobný seznam součástek lze vyčíst v tab. 3.1.

Tab. 3.1. – Seznam součástek

Počet součástek	Název	Hodnota
2	Rezistor	10kΩ
1	Rezistor	330Ω
1	Fotorezistor	-
1	Led dioda bílá	-
7	Propojovací drát	-
1	Arduino Uno	-

Jelikož Arduino Uno využívá logiku 2 V–5 V pro snímání veličin, je zapotřebí převést jiné napěťové logiky do tohoto napětí. Pokud by napěťová hodnota přesáhla 5 V, Arduino Uno deska by byla nenávratně poškozena a nevhodná pro další užití.

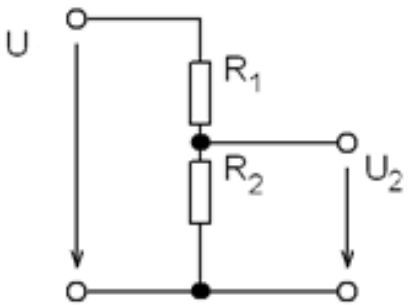


Obr. 3.1 – Schéma zapojení

Lze využít několik variant pro převedení napěťové logiky na požadovanou hodnotu 2 V–5 V. V řešení byl zvolen napěťový dělič. Ale může být použit převodník napětí, který by plnil funkci zcela přesně, pouze s rozdílem, že by bylo možné napětí zvyšovat a snižovat. Schéma eletrického zapojení je uvedeno na obr. 3.1.

Napěťový dělič je obvod složený ze dvou rezistorů, který umožňuje získat konstantní napětí mezi těmito rezistory. Schéma napěťového děliče je znázorněno na obr.3.2. Toto napětí bude vždy menší než je napájecí napětí. Výsledné napětí lze vypočítat následujícím vztahem :

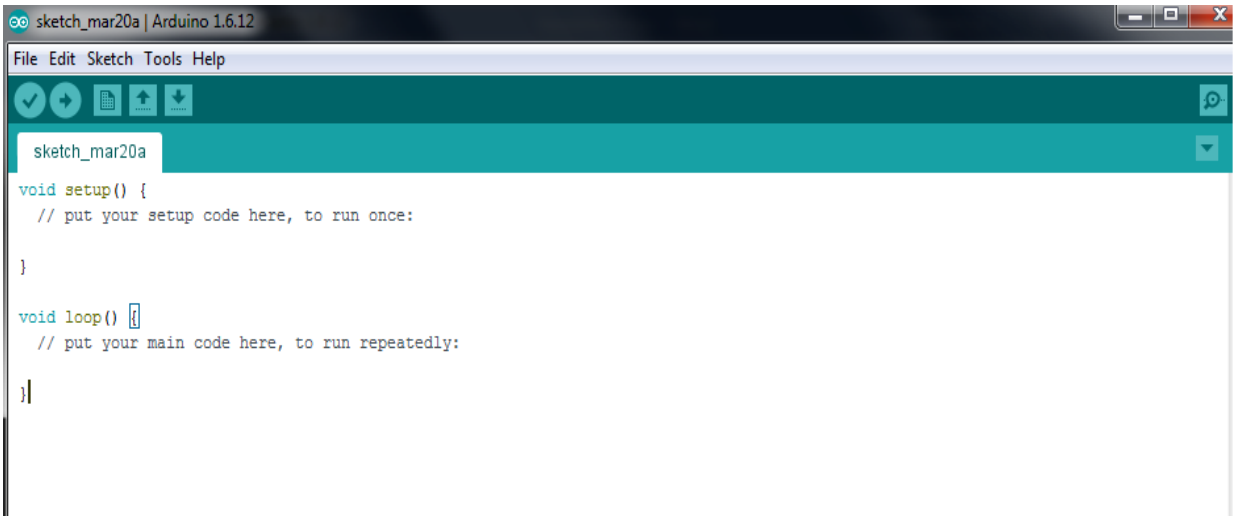
$$U_2 = \frac{R_2}{R_1 + R_2} \cdot U \, .$$



Obr. 3.2 – Napěťový dělič

3.1.2 Realizace programu řízeného procesu

Jak již bylo zmíněno, Arduino desky využívají k programování vlastní vývojový software Arduino IDE. Tento software je dostupný z webových stránek Arduina. Arduino IDE je pod licencí *open-source*. Pro potřeby této práce byla stažena verze 1.8.1 pro operační systém *Windows*. Ke stažení je dostupná řada odkazů. Stáhnout lze rovnou *Windows Installer*, který nainstaluje Arduino IDE přímo, nebo je možné si stáhnout soubor *.zip*, který je určen pro uživatele bez administrátorských povolení. Bezprostředně po instalaci Arduino IDE je



Obr. 3.3 – Arduino IDE

možné jej otevřít pomocí Arduino.exe. Samotné vývojové prostředí je jednoduché a intuitivní jak lze spatřit na obr 3.3.

Arduino IDE je rozděleno na dvě základní části. Funkce *void setup()* představuje část kódu, kde se nastavují všechny parametry pro Arduino desku nebo to, co je potřeba provést pouze jednou při restartu. V této bakalářské práci jsou ve *void setup()* deklarovány použité vstupní a výstupní piny a popřípadě nastavení jejich hodnot. Dále je zde umístěn *Serial.begin(9600)* jehož funkce je popsána v kapitole 2.3.1. Pak je zde umístěno nastavení přerušení pomocí stavových registrů *TCCR1A* a *TCCR1B*. Obsluha přerušení je řešena pomocí vektoru přetečení ve funkci *ICR(TIMER1_OVR_vect){},* ve které je vykonáván program v případě přerušení. Tento program čte hodnotu pomocí *PWM* modulace z pinu pojmenovaném *lightPin* a následně ji zapisuje do sériové komunikace pomocí příkazu *Serial.println()*. Program vepsaný ve funkci *void setup()* je k vidění na obr. 3.4.

```
// proces setup se spustí jednou, jakmile zmáčknete reset
void setup() {
    // nastaví pin 3, 13 jako výstup:
    pinMode(led, OUTPUT);
    pinMode(led_test, OUTPUT);
    // Zapsani log 0 na pin 13
    digitalWrite(led_test, LOW);
    Serial.begin(9600);  //Start seriove komunikace

  noInterrupts();           // vypnuti vsech preruseni
  TCCR1A = 0;               // nastaveni prerusovaciho registru
  TCCR1B = 0;               // nastaveni prerusovaciho registru

  timer1_counter = 3035;

  TCNT1 = timer1_counter;   // preload timer
  TCCR1B |= (1 << CS12);    // 256 prescaler
  TIMSK1 |= (1 << TOIE1);   // povoleni preruseni pri preteceni
  interrupts();             // povoleni preruseni globalne
}

ISR(TIMER1_OVF_vect)        // obsluha preruseni
{
  TCNT1 = timer1_counter;   // casovac preruseni
       int value = analogRead(lightPin);
      Serial.println(value);
     // delay(1000);
}
```

Obr. 3.4 – void setup()

Funkce *void loop()* (obr. 3.5) je funkce, ve které je umístěn hlavní program. Tento program je prováděn opakovaně po dobu přívodu napájení nebo dokud není spuštěna obsluha přerušení. V této funkci je umístěno čtení sériové komunikace pomocí jedné podmínky *if*. Zde se testuje, zda je dostupná sériová komunikace pomocí *Serial.available()*. Je-li sériová komunikace dostupná pro zařízení Arduino, splňuje podmínku a přečte se hodnota na vstupu sériové komunikace. Samotný vstup je převeden na datový typ *Integer* za použití funkce *Serial.parseInt()*. Ten se následně uloží do proměnné *brightness*. *Brightness* se pomocí *PWM* modulace zapíše na výstup pinu *led*.

```
// loop se bude neustále opakovat
void loop() {
    //Precteni bufferu
      if (Serial.available())
      {
      brightness = Serial.parseInt();
      analogWrite(led, brightness);
      }
      digitalWrite(led_test, HIGH);
      //delay(500);

}
```

Obr. 3.5 – void loop()

Na začátku programu byly deklarovány jednotlivé proměnné pro lepší přehlednost programu, jako tomu je zvykem u jazyka *C*. První slovo je vždy datový typ. Ty mohou být numerické, jako například *Integer*, *Double, Float*, nebo mohou pracovat se slovy a znaky. Příkladem znakových datových typů je *String*. Druhé slovo je název proměnné a pak přiřazená hodnota, viz obr. 3.6.

```
int led = 3; // pin, ke kterému je LED připojena
int led_test = 13; // testovaci LED dioda
int brightness = 255; // jas LED diody
int lightPin = 0; //
//Nastaveni preruseni
int timer1_counter;
```

Obr. 3.6 – Deklarace proměnných

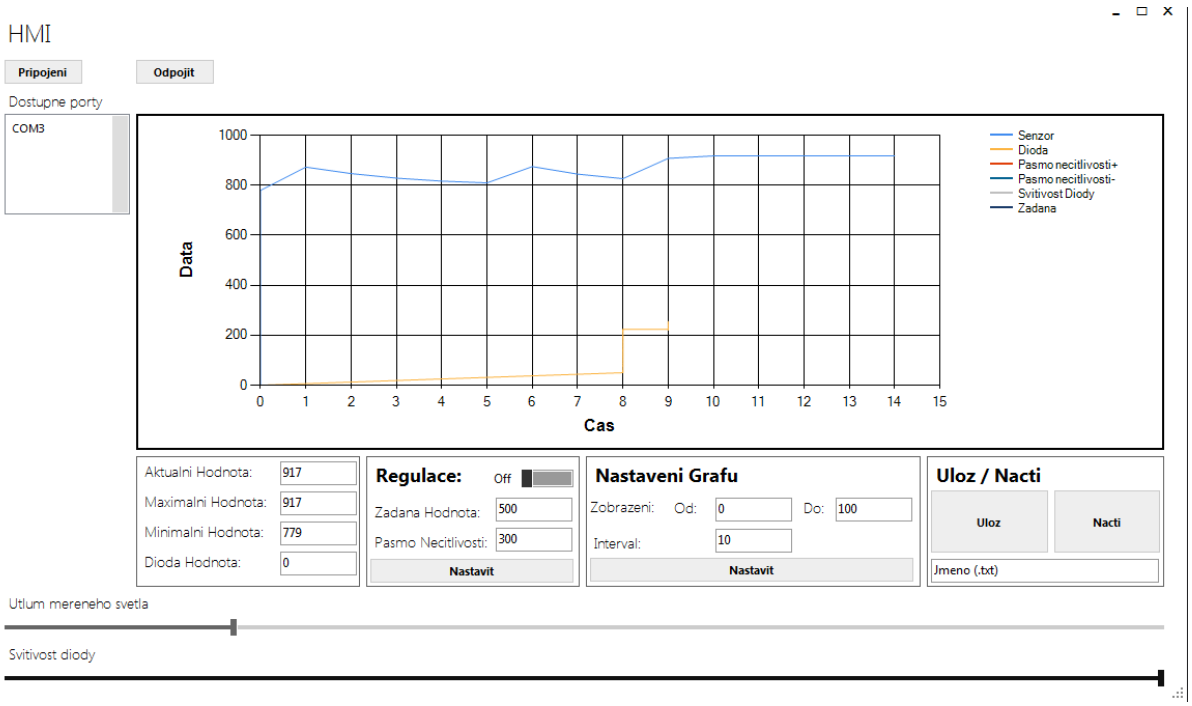
3.1.3 Regulace řízeného procesu

Regulace osvětlení je jednoduchá a nevyžaduje použití složitých druhů regulací. Změny v regulaci se projevují téměř okamžitě. Regulace se skládá se z dvoupolohové regulace a z regulace kde se postupně mění akční veličina. K určení hranic dvoupolohové regulace je využito pásmo necitlivosti, které může být libovolně nastaveno z HMI panelu. Po překročení hraniční hodnoty se k akční veličině přičítá nebo odečítá předem definovaná konstanta, v každém okamžiku měření, až do doby, než se měřená veličina dostane zpět do pásma necitlivosti. V pásmu necitlivosti regulace neprovádí žádné zásahy do akční veličiny.

3.2 REALIZACE HMI APLIKACE

Aplikace byla vytvořena s důrazem na jednoduchost ovládání řízeného procesu. Aby aplikace byla i vizuálně vzhledná, byl použit *MetroFramework*, který se vzhledově podobá operačnímu systému *Windows 8*.

Celou logickou část programu obstarává implementace jedné třídy. V aplikaci lze nalézt zobrazovací graf, ze kterého se dají odečítat hodnoty pomocí kurzoru myši. Dále HMI panel obsahuje seznam dostupných portů, tlačítko *Připojit* na zahájení sériové komunikace a *Odpojit* pro ukončení sériové komunikace. Jsou zde dva posuvníky. Jeden slouží k nastavení svítivosti diody a druhý k simulaci útlumu osvětlení v místnosti. Dále jsou zde umístěny 4 uživatelské panely. První panel slouží k odečítaní naměřených a nastavených hodnot. V druhém panelu je umístěno jednoduché ovládání regulace. Třetí panel obstarává zobrazovací schopnosti grafu. V posledním panelu jsou umístěna dvě tlačítka pro ukládání naměřených hodnot a jejich následné načítaní z textového souboru. Podrobné rozložení je vyobrazeno na obr. 3.7.

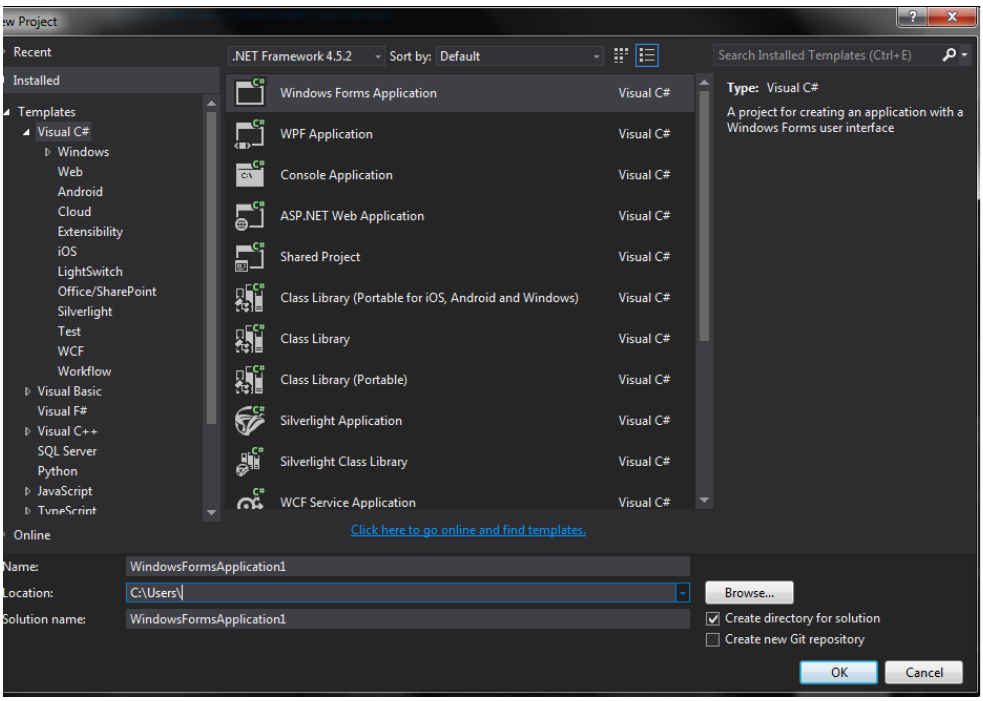


Obr. 3.7 – Jednoduchý HMI panel

3.2.1 Implementace Metro Frameworku do Visual Studia

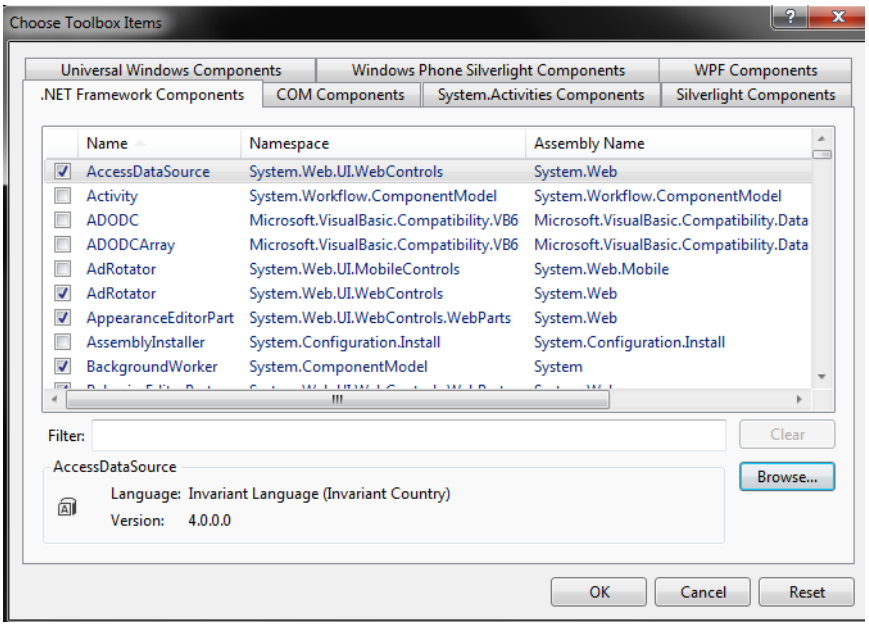
Implementace *MetroFramevorku* do Visual Studia se provádí pomocí importu knihoven DLL. V této práci jsou použity knihovny stažené a volně dostupné z stránky *GitHub.com*. Na této stránce je potřebná registrace, po níž uživatel získá přístup k souborům, které sdílí další uživatelé. Po prohledání byl stažen soubor ve formátu *.zip* se jménem *Winforms Modern UI Metro Framework*. Po stažení knihovny je třeba rozbalit soubory do složky.

K implementaci je třeba založit novou *Windows Forms Application*. Po otevření Visual Studia je nutné vybrat *New Project*, kde se z nabídky po pravé straně vybere *Visual C#*, *Windows* a zde zvolit *Windows Forms Application*. V dolním poli je třeba zadat jméno a umístění nově vytvořených souborů. Tvorba *Windows Forms Application* je znázorněna na obr. 3.8.



Obr. 3.8 – New Project v prostředí Visual Studio

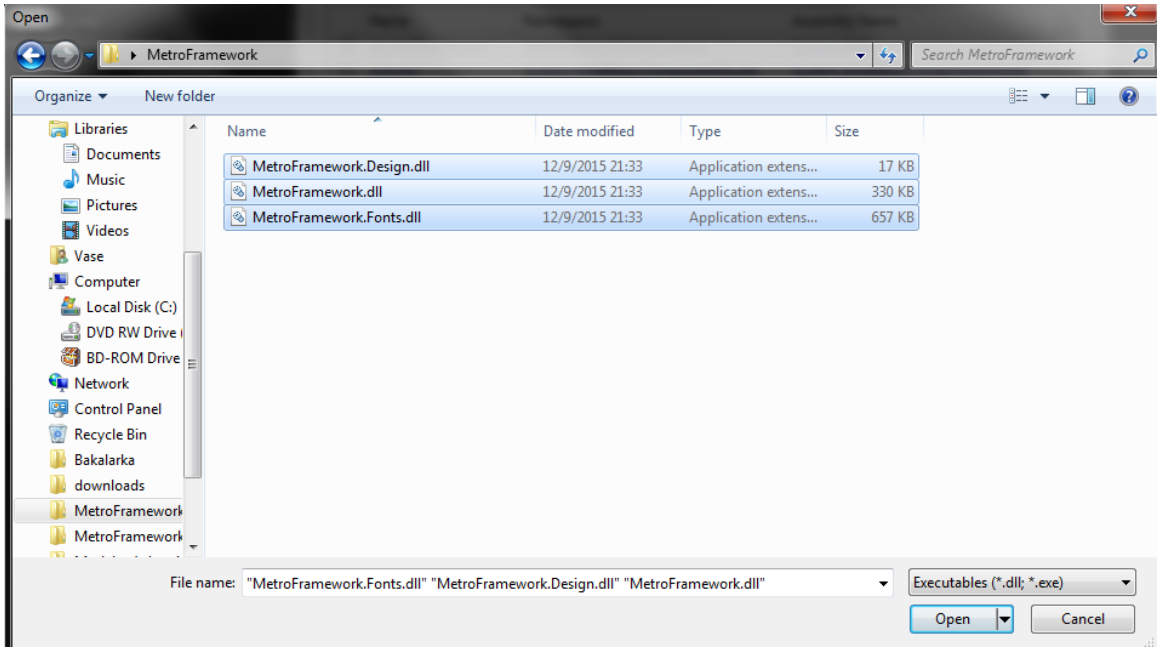
Ve vytvořeném novém projektu otevřeme záložku *Tools* a následně vybereme *Choose Toolbox Items*.



Obr. 3.9 – Choose Toolbox Items v Visual Studio

Objeví se okno jako na obr. 3.9, dále je třeba vybrat tlačítko *Browse* a zadat cestu do složky, kde je rozbalený soubor *Modern UI Metro Framework*. Zde se zvolí *MetroFramework.dll*.

Po přidání je do projektu nutné dodat reference na *MetroFramework.dll*, *MetroFramework.Desing.dll*, *MetroFramework.Fonts.dll*. (obr. 3.10). V pravé části projektu



Obr. 3.10 – Přidání MetroFramwork do Visual Studio

se nachází *Solution explorer,* ve kterém je umístěna složka *References*. Složku je třeba označit a pravým tlačítkem vybrat *Add Refere*nces a následně přidat celý obsah složky *Modern UI Metro Framework.* Nyní je možné pracovat s MetroFrameworkem ve Visual Studio.

3.2.2 Program HMI Aplikace

Aplikace je implementována pomocí jedné třídy. Obsahuje 17 metod, které obstarávají logiku programu. Na začátku je několik pomocných proměnných, tyto proměnné jsou vysvětleny v používaných metodách.

- *Private void hledejPorty*()*:* Tato metoda načte pomocí funkce *GetPortNames*() seznam všech dostupných portů, které následně podmínkou *foreach* přidá a zapíše jejich jména do listBoxu *listPort*. Ten je umístěn v levém horním rohu aplikace, označen jako *Dostupné porty.* Jestli počet dostupných portů je roven nule, zobrazí se chybová zpráva o neúspěšném hledání portů.
- *Private void Form1_Load (object sender, EventArgs e)*: Metoda Form1_Load se spouští při správném nahrání a zobrazení aplikace. Tato metoda obsahuje volání na další dvě metody, a to *hledejPorty*() a *vytvorGraf*().
- *Private void vytvorGraf()*: vytvorGraf je relativně rozsáhlá metoda starající se o nastavení a správné zobrazení grafu samotného. Nejprve je nutné vytvořit série hodnot, které se budou v grafu zobrazovat. To se provede funkcí *Series.Add(*"Jmeno serie"*).* Definice jména série řeší *Series.Name("Jmeno Serie")* a požadovaný typ grafu, liniový, *Series["Jmeno Serie "].ChartType = SeriesChartType.Line;*. Dále se v této metodě definuje plocha pod samotnými sériemi. Oblast grafu se přiřazuje samostatně pro osu X a pro osu Y, pro které se nastavuje jejich barva, popis a font os, barva popisu os, maximální a minimální zobrazovací rozsah. Pak se zde pro každou osu globálně povolují uživatelské operace, jako například automatické rolování, zoomování a výběr plochy označené uživatelem samotným.
- *private void graf_DoubleClick(object sender, EventArgs e)*: Zde se provádí samostatné zoomování grafu pomocí akce dvojitého poklepání na oblast grafu.
- *private void graf_MouseMove(object sender, MouseEventArgs e)*: K odečítání hodnot z grafu se používá *Series[].Tooltrip* funkce. Za touto funkcí bude následovat formát, v jakém se hodnoty budou zobrazovat. Metoda využívá akci pohybu myši po vybraném objektu.

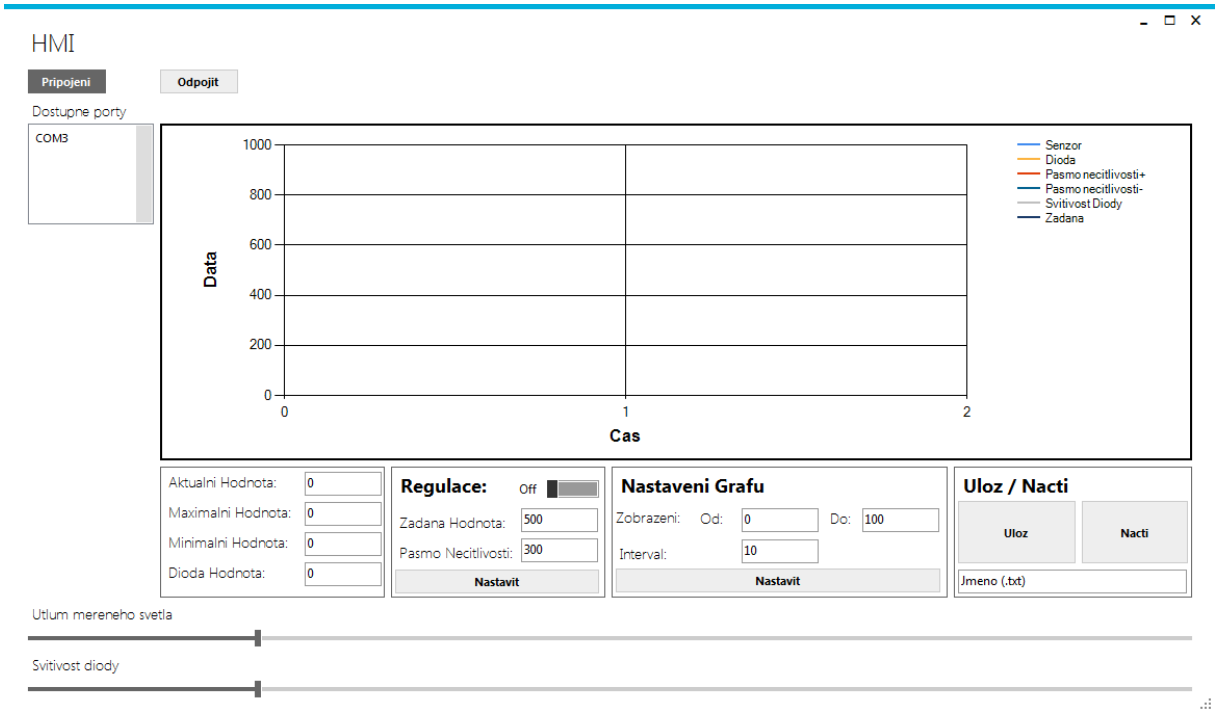
- *private void pripojeniBtn_Click(object sender, EventArgs e):* Zde se volá metoda *NastavSerial(),* ta dále obsluhuje nastavení sériové komunikace. Pokud nastane situace, kdy je připojeno k zařízení více sériových portů, je uživatel nucen si vybrat jen jeden.
- *private void NastavSerial()*: Tato metoda nastavuje sériovou komunikaci. Vytváří nový virtuální sériový port, který je nastaven na rychlost komunikace 9 600 bitů za sekundu, je vypnuta paritní kontrola bajtů a jeden bajt je definován na 8 datových bitů. Pomocí výjimky se zde testuje spojení PC a Arduina. Pokud je výsledek negativní, tak se zobrazí chybová hláška s textem „Error". Dále se zde vytváří delegát, který odkazuje na obsluhu čtení ze sériové linky.
- *private void myPort_DataReceived(object sender, SerialDataReceivedEventArgs e)*: V této metodě se pomocí *ReadLine()* čte ze sériové linky celý řádek a odvolává se zde na ovladač událostí definující další postup v případě přečtení celého řádku ze sériové linky.
- *private void dislayData_eventHandler(object sender, EventArgs e)*: Toto je ovladač událostí volaný v *myPort_DataReceived.* Zde se zapisuje aktuální naměřená hodnota do příslušného textového pole a poté se převádí z datového typu *string* na datový typ *Int32,* aby bylo možné hodnotu zaznamenat do grafu jako číslo. Naměřená hodnota se navíc zapisuje do dvourozměrného pole datového typu *Int,* které se dále používá při ukládání dat do textového souboru. Následně je zde volána metoda *hledejMaxMin(Int)* a podmínka o zapnutí regulace. Jestli je vyhodnocení podmínky úspěšné, zavolá se metoda *regulace(Int).* Neúspěšné vyhodnocení podmínky znamená, že aplikace na vstup sériové komunikace zapíše hodnotu ovládající svítivost diody. Ta je nastavena uživatelem pomocí posuvníku, označeném jako *Svítivost Diody*.
- *private void regulace(int i):* Jak lze odvodit již z názvu, tato metoda se zabývá regulací akční veličiny. Do zobrazeného grafu funkce přidává čtyři nové série dat, *Pasmo necitlivosti+, Pasmo necitlivosti-, Zadana,* a *Svitivost Diody.* Celková velikost pásma necitlivosti je rovna součtu hodnot *Pasmo necitlivosti+* a *Pasmo necitlivosti-* a pohybuje se okolo žádané hodnoty, jak v kladném, tak v záporném směru. Proměnná *Zadana* je regulovaná veličina. Samotná regulace probíhá pomocí dvou podmínek. První podmínka testuje, zda je žádaná veličina nad pásmem necitlivosti a pokud ano, sníží svítivost diody o požadovanou hodnotu a zapíše jí na sériovou linku. Druhá podmínka je podobná jako ta první, s tím rozdílem, že vyhodnocuje hodnotu nižší, než

je pásmo necitlivosti. Jestli je druhá podmínka vyhodnocena kladně, dojde ke zvýšení svítivosti diody o požadovanou hodnotu.

- *private void hledejMaxMin(int i):* V metodě se testuje, zda je aktuální měřená hodnota větší nebo menší než předcházející maximální nebo minimální hodnota. Toto testování probíhá za přispění dvou podmínek *if*, jedné pro maximální a jedné pro minimální hodnotu. Pak tyto hodnoty převádí do datového typu *string* a zapisuje je do příslušných textových polí v aplikaci.
- *private void metroTrackBar1_Scroll(object sender, ScrollEventArgs e):* Tato metoda obsluhuje posuvník, kterým se ručně nastavuje svítivost diody. A to pouze v případě, že je splněna výjimka a sériový port je otevřený. Jinak se zobrazí chybová hláška s danou chybou.
- *private void metroToggle1_CheckedChanged(object sender, EventArgs e):* Přepínač, který zapíná nebo vypíná možnosti regulace pouze v případě, že jsou regulační hodnoty řádně vyplněny. Musí být vyplněna žádaná hodnota a rozsah pásma necitlivosti.
- *private void nastavitBtn_Click(object sender, EventArgs e):* Metoda obsluhuje akci kliku na tlačítko *Nastavit*, kterým se nastavují hodnoty regulačních veličin. Tyto veličiny jsou Žádaná hodnota a pásmo necitlivosti.
- *private void nastGrafBtn_Click(object sender, EventArgs e)*: K nastavení zobrazovacích schopností grafu se používá tlačítko *Nastavit*, které obsluhuje tato metoda, v panelu Nastaveni Grafu. Metoda sbírá údaje z textových polí, které zobrazí konkrétní data z časové osy x, v rozmezí od a do. Dále je zde použit interval, který upraví zobrazení hustoty svislých čar na mřížce pod grafem.
- *private void odpojitBtn_Click(object sender, EventArgs e)*: Tato metoda obsluhuje tlačítko odpojit. Používá k tomu výjimku testující to, zda funkce *.Close()*, proběhla úspěšně. *Close()* řádně ukončuje sériovou komunikaci ze strany PC. Pokud sériová komunikace není ukončena, zobrazí se chybová hláška s daným problémem.
- *private void UlozBtn_Click(object sender, EventArgs e)*: Akce tlačítka uložit obsažená v této metodě ukládá naměřené hodnoty spolu s časem jejich měření do textového souboru. Data a čas jsou uloženy v poli, které se následně pomocí abstraktní třídy *StreamWriter* uloží řádek po řádku do souboru s příponou .txt. Aby se předešlo chybám při zápisu, je i zde umístěna výjimka, která testuje správnost zapisování a v případě chyby o ní informuje pomocí chybové hlášky.

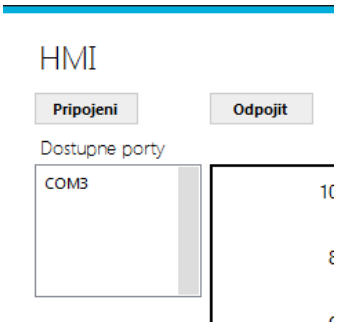
3.3 UKÁZKA ČINOSTI HMI APLIKCE

Jak je patrné z obr. 3.11, při prvním spuštění aplikace se otevře okno se všemi ovládacími prvky aplikace. Pokud se aplikace nespustí a zobrazí se chybová hláška, tak je nutné nejdříve odstranit závadu popsanou v chybové hlášce a pak aplikaci znovu spustit. Oknu aplikace vévodí velké pole grafu, kde se zatím nenačítají žádné hodnoty.



Obr. 3.11 – Aplikace bez hodnot

Pro načtení a zobrazení některých hodnot je nutné vybrat ze seznamu sériových portů, který je umístěn v levé horní části okna, příslušný port, který chceme zobrazovat a regulovat. Po vybrání portu lze kliknout na tlačítko *Pripojeni,* čímž se otevře sériová linka mezi PC a zařízením Arduino Uno. Sériovou komunikaci lze ukončit pomocí tlačítka *Odpojit* umístěného vedle tlačítka *Pripojeni*. Tlačítko *Odpojit* a *Pripojeni* vykresluje obr. 3.12.

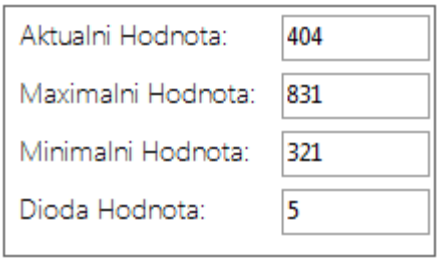


Obr. 3.12 – Ovládání sériové komunikace

Po připojení zařízení se automaticky začnou zobrazovat přijatá data do grafu a do prvního panelu pod grafem. V tomto panelu jsou čtyři hodnoty, které nejdou nastavovat, a to:

- *Aktualni Hodnota,* která zobrazuje právě přijatou hodnotu z čidla,
- *Maximalni Hodnota*, kde je uvedena maximální dosažená hodnota,
- *Minimalni Hodnota*, kde je uvedena minimální naměřená hodnota
- *Dioda Hodnota,* která ukazuje, jak jasně svíti zapojená dioda.

Ukázka hodnot je vidět na obr. 3.13.

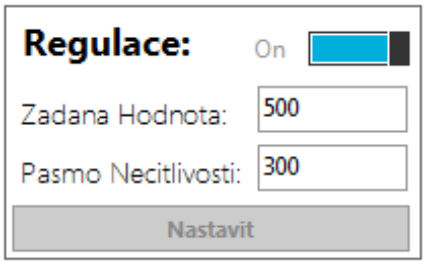


Obr. 3.13 – Naměřené hodnoty

Další panel je regulační umožňující nastavení regulace. Pomocí okna *Zadana Hodnota* lze zadat hodnotu, na kterou se má žádaná veličina regulovat. Pod tímto oknem je *Pasmo Necitlivosti,* ve kterém zadáváme tolerované pásmo necitlivosti okolo žádané hodnoty. Po

zapsání hodnot je nutné kliknout na tlačítko *Nastavit* v daném panelu, k ověření správnosti zadaných dat. Pak je možné pomocí přepínače umístěného v horní části panelu zahájit regulaci. Panel a ukázku nastavení zobrazuje obr. 3.14.

V panelu Nastaveni Grafu lze měnit zobrazený obsah samotného grafu. Pokud je nutné



Obr. 3.14 – Panel Regulace

vybrat z časové osy informace z určitého časového úseku, lze toho docílit změnami v textových polích na řádku *Zobrazeni*. Tato pole se jmenují *Od* a *Do*. Dále je možné nastavit hustotu zobrazovaných hodnot na časové ose pomocí pole *Interval*. Pro změnu zobrazení je opět nutné kliknout na tlačítko *Nastavit* v daném panelu, jak ukazuje obr. 3.15.

Obr. 3.15 – Nastavení grafu

Pomocí panelu *Uloz*/*Nacti* je možné měřený průběh uložit a pak následně načíst. K uložení slouží tlačítko *Uloz* a textové pole umístěné pod ním. Hodnoty se uloží do textového souboru. Po kliknutí na tlačítko *Nacti* se zobrazí průzkumník systému Windows, kde je zapotřebí nalézt požadovaný soubor pro načtení. Panel je zobrazen na obr. 3.16.

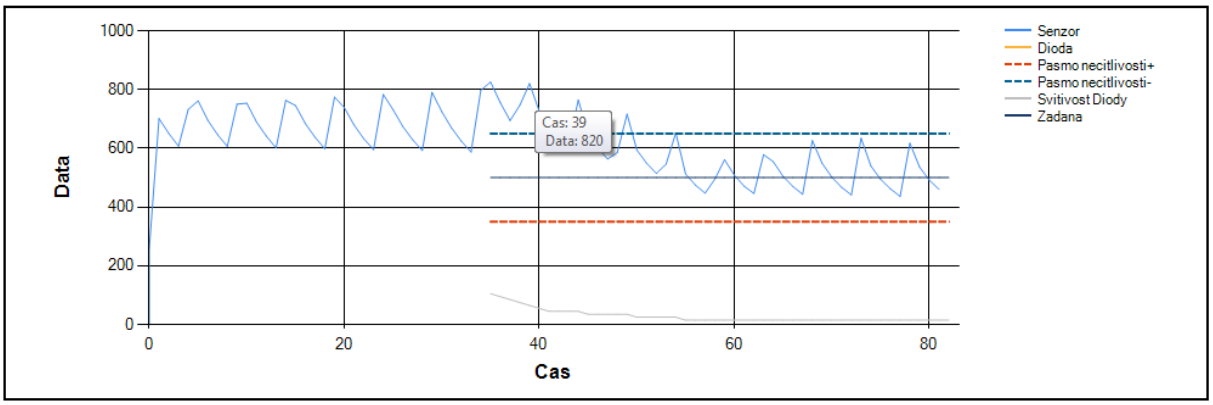
Obr. 3.16 – Ukládání a načítání z .txt

Pod všemi ovládacími panely jsou umístěné dva posuvníky. Tyto posuvníky slouží k ručnímu nastavení jasu diody a simulovanému útlumu měřené veličiny. První posuvník nazvaný *Utlum mereneho svetla*, lze ovládat i při zapnuté regulaci a slouží k simulovanému útlumu okolního světla. Druhý posuvník má název *Svitivost diody*, který je použit k ovládání jasu diody, ale na rozdíl od prvního posuvníku jej není možné ovládat po dobu zapnutí regulace. Oba posuvníky jsou k vidění na obr. 3.17.

Obr. 3.17 – Posuvníky pro ovládání jasu a simulovaného útlumu

Poslední částí aplikace je graf, ve kterém se zobrazují naměřená data. Tento graf se dynamicky mění v závislosti na načítaných datech. V levé části grafu je zobrazena legenda popisující jednotlivé složky grafu. Uživatel má možnost zobrazovat vybraná data pomocí akcí na myši. Pokud najede kurzorem myši na určitou část grafu, tak se mu zobrazí naměřená hodnota a čas, z daného měřeného bodu, jak je zobrazeno na obr. 3.18. Dále je možnost pomocí dvojkliku na levém tlačítku myši přiblížit určitou oblast grafu.

Ovládání aplikace je jednoduché a intuitivní. Uživatel se snadno zorientuje v okně aplikace a najde potřebné informace o měřené a regulované hodnotě. Velkou výhodou je zobrazení určitých hodnot pomocí najetí myši na oblast grafu.



Obr. 3.18 – Graf aplikace

4 ZHODNOCENÍ

Realizovaná aplikace slouží k jednoduchému čtení právě naměřených dat a jejich dalšímu zpracovaní. Řízení regulovaného procesu se zde provádí promocí jednoduché regulace, která je pro tento proces postačující. Jednoduchost zapojení umožňuje snadnou modulárnost a změnu řízeného procesu. Samotná aplikace, která bude zobrazena na HMI panelu, je, co se ovládání týče, velice intuitivní a nedovoluje špatné nastavení parametrů.

5 ZÁVĚR

Dle zadání bakalářské práce bylo vytvořeno HMI rozhraní a současně byl vytvořen řízený proces a řídící systém, který je možné ovládat z HMI rozhraní.

Sestavení řízeného procesu použitím platformy Arduino bylo intuitivní. Deska Arduino Uno, která byla použita při stavbě plnohodnotně vyhovuje účelům této práce. Téměř neomezené možnosti použité desky umožňovaly zvolení libovolného procesu. Aby řízený proces nebyl časově náročný na regulaci a jeho změny se daly měřit okamžitě, bylo zvoleno měření osvětlení namísto ostatních procesů jako je například teplota či vlhkost. Na Arduino desce nebyly využity všechny zapojitelné piny, a tak by se zapojení dalo rozšířit o další měřící nebo akční členy. Využitím Arduino IDE, ve kterém se výsledný program pro desku psal, se stalo programování výrazně jednoduší než v klasickém jazyce C. Práce s Arduinem Uno byla velice zábavnou záležitostí, která má smysl využití i do budoucna díky veliké rozmanitosti celé platformy a běžně dostupným schématům.

Hlavním cílem práce je aplikace psaná v jazyce C# pro operační systém Windows. V ní byla implementována možnost regulace s nastavitelným pásmem necitlivosti. Tato regulace je dostatečně přesná, a tak není potřeba využití složitější regulace např. s PID regulátorem. Využitím aplikace lze snadno odečítat a regulovat požadované hodnoty které se zobrazují v grafu. Jazyk C# má mnoho vnořených funkcí, jejichž pomocí se snadno vytváří grafické rozhraní. Aplikace je tedy snadno rozšiřitelná o další prvky, které by byly požadovány k danému procesu.

POUŽITÁ LITERATURA

Arduino 2WD Robot. 2016. *RobotShop* [online]. [cit. 2017-04-14]. Dostupné z: http://www.robotshop.com/en/arduino-2wd-robot-north-american-plug.html

Arduino.cz: Webový Magazín O Arduinu A Elektronice. 2015 [online]. [cit. 2017-04-28]. Dostupné z: http://arduino.cz/

BĚHÁLEK, M. 2007. Programovací jazyk C#. http://www.cs.vsb.cz [online]. [cit. 2017-04-13]. Dostupné z: http://www.cs.vsb.cz/behalek/vyuka/pcsharp/text/

HANDL, A. 2015. Arduino TRE se brzy ocitne na trhu. Arduino.cz: Webový magazín o Arduinu a elektronice [online]. [cit. 2017-04-13]. Dostupné z: http://arduino.cz/arduino-tre-se-brzy-ocitne-na-trhu/

HMI Guinde 2017. AnaheimAutomation [online]. [cit. 2017-04-14]. Dostupné z: http://www.anaheimautomation.com/manuals/forms/hmi-guide.php#sthash.9fZtPz58.F8ebOi27.dpbs

LEHRBAUM, R. 2013. Arduino TRE SBC runs Linux on TI Sitara AM335x. HackerBoards [online]. [cit. 2017-04-15]. Dostupné z: http://hackerboards.com/arduino-tre-sbc-runs-linux-on-arm/

PETZOLD, CH. 2003 Programování Microsoft Windows v jazyce C#. Praha.: SoftPress, 1208 s. ISBN 80-86497-54-2

Serial 2017. Arduino [online]. [cit. 2017-04-14]. Dostupné z: https://www.arduino.cc/en/Reference/Serial

VALTER, J. 2017. Regulace. valter.byl.cz: Aplikace a software AMiT, poradenská činnost MaR, řídicí systémy a vše co jste chtěli vědět o regulaci [online]. [cit. 2017-04-13]. Dostupné z: https://valter.byl.cz/sites/default/files/soubory/regulace.pdf

VODA, Z. 2015. Průvodce světem Arduina. Bučovice: Martin Stříž, ISBN 978-80-87106-90-7.

PŘÍLOHY

Příklad

A - CD