TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

BAKALÁŘSKÁ PRÁCE

Liberec 2013 **Matěj Kolář**

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2646 – Informační technologie
Studijní obor: 802R007 – Informační technologie

Interaktivní šachový automat z LEGO Mindstorms NXT

Interactive chess computer made of Lego Mindstorms NXT

Bakalářská práce

Autor:	Matěj Kolář
Vedoucí bakalářské práce:	doc. Ing. Josef Chaloupka, Ph.D.
Konzultant:	Ing. Karel Paleček

V Liberci 17. 5. 2013

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum: 17. 5. 2013

Podpis:

Poděkování

Tímto bych chtěl poděkovat vedoucímu bakalářské práce panu doc. Ing. Josefu Chaloupkovi, Ph.D. za užitečné konzultace, rady a věcné připomínky k této práci. Dále mu také děkuji za jeho trpělivost a vstřícnost.

Abstrakt

V rámci této bakalářské práce byl vytvořen šachový automat pomocí interaktivní stavebnice LEGO Mindstorms NXT, ovládaný pomocí obslužné aplikace napsané v MS Visual C++. Systém umožňuje zahrát šachovou partii mezi člověkem a počítačem. Po zadání tahu uživatelem do grafické aplikace je ověřena správnost pomocí šachového programu s vnitřní reprezentací herního pole. V případě neplatného tahu je uživatel upozorněn a vyzván k zadání nového tahu. Jinak je tah zpracován šachovým programem a je započato vyhledávání tahu protihráče. Nakonec je tah uskutečněn na hracím poli pomocí automatu a graficky vykreslen.

Obslužná aplikace obsahuje grafické prostředí pro vykreslení herního pole a figurek hry šachy. Grafická část aplikace má strukturu formulářového okna a je napsána pomocí jazyku C++/CLI. Herní pole je v aplikaci uloženo jako dvourozměrné pole o velikosti 8*8 polí a každé pole obsahuje odkaz na třídu reprezentující danou figurku. Další součástí aplikace je vlastní struktura pro spuštění a komunikaci se šachovým programem Crafty. Tento program je spuštěn ve vlastním procesu a pro komunikaci s ním je použito přesměrování standardního vstupu a výstupu. Přesměrovaná data jsou zapisována a čtena asynchronně, proto bylo nutné vytvořit datovou strukturu používající více vláken pro zpracování těchto dat. Komunikace se šachovým programem probíhá pomocí použití protokolu CECP a vlastních metod zpracovávajících příkazy z programu nebo aplikace.

Pro konstrukci automatu byla použita mechatronická stavebnice Lego Mindstorms NXT. Systém využívá souřadnicový systém dvou os, poháněný dvojicí servomotorů. Měření vzdáleností a kalibraci zajišťuje senzor otáček v motorech a dvojice dotykových senzorů na každé ose. Figurka je posunována pomocí elektromagnetu připevněného k pohyblivé konstrukci automatu.

Automat je ovládán pomocí aplikace s knihovnou NXT++, která obsahuje nástroje pro ovládání systému NXT.

Klíčová slova: šachový automat, lego, mindstorms, nxt, C++/CLI, formulářová aplikace

Abstract

As a part of this bachelor's project, a chess machine has been created using a programmable robotics kit called LEGO Mindstorms NXT, controlled through an application written in MS Visual C++. This system enables the user to play a chess match against an AI-controlled opponent. After a move is entered into a graphics environment, the application will check its correctness, utilising a chess program with its own model of a chess-board. In case of an invalid move, the user is asked to enter a new move. In other case the move is processed and the program begins searching for the opponent's move. Afterwards, the move is executed by the chess machine and displayed in the graphics environment.

The application includes a graphics environment for displaying the playing field and pieces used in chess. The graphics environment has the structure of a form window and is written in C++/CLI. The playing field is stored as a two dimensional field with a size of 8*8 squares and each square contains a reference to a class representing a given chess piece.

The next part of the application is a structure for lauching and communicating with the chess program Crafty. This program is launched as a new process and is communicated with through redirecting the standard input and output. The redirected data are written and read asynchronously, so it was necessary to create a data structure utilising several threads for processing given data. Communication with the chess program is done through the CECP protocol and various custom methods.

The chess machine was constructed using a programmable robotics kit called LEGO Mindstorms NXT. The system uses a grid with two axes, run by two servo motors. Measurement of distances and calibration are done by in-built rotary encoders and a pair of touch sensors on each axis. A chess piece is moved using an electromagnet attached to the chess machine.

The chess machine is controlled through an application containing an NXT++ library, which contains tools for controlling the NXT system.

Key words: chess machine, LEGO, Mindstorms, NXT, C++/CLI, form application

Obsah

Seznam tabulek

Seznam ilustrací

Seznam použitých zkratek

CECP – Chess Engine Communication Protocol

CLR – Common Language Runtime

CLI – Common Language Infrastructure

C++ - Programovací jazyk C++

.NET – Microsoft .NET Framework

UCI – Universal Chess Interface

USB – Universal Serial Bus

NXT – Systém Lego Mindstorms NXT

Úvod

V současné době se začínají čím dál, tím více používat mechatronické stavebnice pro účely výuky. Tyto systémy umožňují seznámit člověka se základními problémy konstrukce a programování. S jejich pomocí je možné vytvořit jednoduchou konstrukci ovládanou přímo z inteligentní kostky stavebnice nebo je možné sestavit sofistikovaný systém s vícero kostkami, ovládáný pomocí počítače. Mezi hlavní přednosti patří zejména obsáhlost a dostupnost informací na internetu, kde je možné nalézt postupy konstrukcí a možnosti programování. Použití stavebnice je jednoduché a její stavbu zvládne i méně zkušený člověk. Ani programování není složíté a v případě použití jednoduchých funkčních bloků lze bezproblému sestavit zajímavý systém. Tyto přednosti dělají ze stavebnice základní pomůcku při podnícení zájmu dětí a lidí o elektrotechniku a možnost rozšíření systému vede k dalšímu zájmu o problematiku těchto systémů.

Šachový automat má v dnešní době celou řadu využití. Lze ho použít k nácviku programovacích algoritmů pro posun figurky nebo si vyzkoušet použití konstrukčních prvků stavebnice. Funkční automat lze dále využít jako pomůcku při hře šachy, kdy je uživatelem ovládán a protitah je generován pomocí šachového programu. Takto může uživatel trénovat své dovednosti ve hře. Díky automatizovanému posunu figurek je možné systém použít pro účel hry tělesně handicapovaných lidí, kdy je posun zadáván v textovém režimu pomocí algebraického výrazu. V kombinaci se zvukovým rozpoznáním řeči se tak stává použití a hra snadnou aktivitou.

Další předností systému je implementace šachového programu pro vyhledání optimálního tahu počítače. Vzhledem k použití protokolu pro komunikaci s programem ho je možné vyměnit za jiný program, který používá stejný protokol. Tato vlastnost se dá využít pro otestování či porovnání více šachových programů. Navržený systém je velice komplexní a jeho použití se stává jednoduché. Posun figurek dobře znázorňuje stav hry šachy a pro množství uživatelů je tato vizuální reprezentace o mnoho přívětivější než pouhé grafické znázornění na počítači.

1. Teoretická část

1.1 Mechatronická stavebnice Lego Mindstorms NXT

Lego Mindstorms je celosvětově známá interaktivní stavebnice vyráběná firmou Lego. V dnešní době se používá hlavně jako výuková a zábavní pomůcka pro seznámení se se základy konstrukce a programování. Pro konstrukci nepohyblivých částí se používá standardních součástek Lego Technics. Systém dále obsahuje inteligentní kostku, která slouží jako hlavní článek a jsou k ní připojeny až 3 výstupní a 4 vstupní zařízení. Pomocí této stavebnice je možné sestavit a naprogramovat složité roboty nebo automaty. [1]

Obr. 1: Lego Mindstorms NXT

1.1.1 Vstupní zařízení systému NXT

Systém Lego Mindstorms nabízí pro vstupní zařízení celkem 4 porty. Na vstupní port lze připojit vždy jen jeden senzor. Systém umožňuje připojení základních senzorů čtyr typů, kterými jsou: senzor světla, senzor zvuku, senzor doteku a ultrazvukový senzor. Všechny tyto senzory je možné využít k měření fyzikálních veličin a následně naměřené hodnoty zpracovat pomocí inteligentní kostky NXT.

- **Dotykový senzor** – Nejčastěji se dotykový senzor používá jako tlačítko. Jeho hlavní funkcí je zjistit, zda je stlačen nebo uvolněn. Na základě toho, v jakém se nachází stavu, odesílá na vstupní port logickou hodnotu pravda nebo nepravda. Využití najde například jako dojezdová zarážka pro servomotor nebo tlakový spínač pro zjištění přítomnosti nějakého předmětu.

Obr. 2: Dotykový senzor NXT

- **Ultrazvukový senzor –** Používá se ke zjištění vzdálenosti od měřeného předmětu. Jeho hlavní funkcí je mapování okolního prostředí na základě vysílání ultrazvukové vlny a měření, za jakou dobu se vlna odrazí a vrátí zpět k přijímači. Rozsah přesnosti měření je od 0 do 255 centimetrů.

Obr. 3: Ultrazvukový senzor NXT

- **Zvukový senzor –** Měří hladinu zvuku v okolí. Naměřenou hodnotu udává v jednotkách dB, jsou měřeny zvuky vysokých i nízkých frekvencí.

Obr. 4: Zvukový senzor NXT

- **Světelný senzor –** Pomocí světelného senzoru lze rozeznávat světlo, tmu a také intenzitu světla. Využití nachází hlavně při orientaci a navigaci robota.

Obr. 5: Světelný senzor NXT

1.1.2 Výstupní zařízení systému NXT

Výstupní zařízení systému NXT slouží k indikaci naměřených veličin a možnosti realizace pohybu částí robota. Nejpoužívanější výstupní zařízení je krokový elektromotor. K připojení výstupních zařízení jsou k dispozici 3 porty.

Servomotor systému NXT je složen z elektromotoru a otáčkoměru. Elektromotor zajišťuje plynulé otáčení servomotoru a otáčkoměr měří pomocí optických čidel míru natočení. Otáčkoměr je schopen měřit s přesností jednoho stupně. Servomotor je možné otočit o zvolený počet stupňů nebo počet sekund. Rychlost otáčení je možné nastavit od 0% do 100% maximální rychlosti.

Obr. 6: Servomotor systému NXT

1.1.3 Inteligentní kostka NXT

Řídicí jednotka nebo také inteligentní kostka systému NXT je základním stavebním kamenem pro celý systém. Obsahuje hlavní řídicí modul pro komunikaci se vstupními a výstupními obvody. Pro komunikaci s ostatními zařízeními využívá rozhraní USB nebo Bluetooth. Dále obsahuje 4 vstupní a 3 výstupní porty pro připojení externích modulů systému NXT.

Obr. 7: Inteligentní kostka NXT

1.1.4 Možnosti programování systému NXT

Systém Lego Mindstorms NXT je možné programovat dvěma způsoby. Prvním je programování přímo v inteligentní kostce. Tento způsob je vhodný pro nejjednodušší programy, protože zde není na výběr mnoho funkčních bloků a možnosti programování jsou tedy značně omezené. Druhým způsobem je programování systému přes některé z komunikačních rozhraní. Na výběr je rozhraní USB nebo Bluetooth. V tomto případě je systém ovládán externím programem, který zpracovává data a požadavky a do systému odesílá pouze příslušné příkazy pro ovládání. Pro ovládání systému NXT v některém z programovacích jazyků je nutné použít externí knihovnu. Tyto knihovny jsou dostupné pro vetšinu dnes používaných jazyků. [2]

Název knihovny	**Programovací jazyk**
brickOS	C/C++
LabVIEW	LabVIEW
Lego::NXT	Perl
leJOS	Java
nxtOSEK	C/C++
NXT++	C++
NXT_Python	Python
RWTH – Mindstorms NXT Toolbox	MATLAB
NXT.NET	C#

Tabulka 1: Knihovny pro systém NXT

Pro programovací jazyk C++ existuje více knihoven. Pro tento systém byla vybrána knihovna NXT++, protože obsahuje všechny potřebné funkce pro ovládání robota v dostatečném rozsahu pro tuto aplikaci. Tato knihovna se skládá z několika hlavičkových a zdrojových souborů a obsahuje také knihovnu Fantom, která je nezbytná pro ovládání systému NXT. Pro správnou funkčnost v programovacím jazyce C++/CLI je nutné přidat všechny hlavičkové a zdrojové soubory do projektu a následně nastavit kompilátor na základní kompilátor pro C++/CLI.

Použité funkce z knihovny NXT++:

- *NXT::OpenNXTDevice* – pro vytvoření spojení s kostkou NXT
- *NXT::Sensor::SetTouch* – pro nastavení senzoru doteku.

- *NXT::Sensor::GetValue* – pro vyčtení stavu senzoru
- *NXT::Motor::ResetRotationCount* – pro vyresetování úhlu natočení motoru
- *NXT::Motor::GetRotationCount* – pro vyčtení úhlu natočení motoru
- *NXT::Motor::SetForward* – pro otáčení motoru dopředu
- *NXT::Motor::Stop* – pro zastavení otáčení motoru
- *NXT::Motor::GoTo* – pro otočení motoru o požadovanou hodnotu

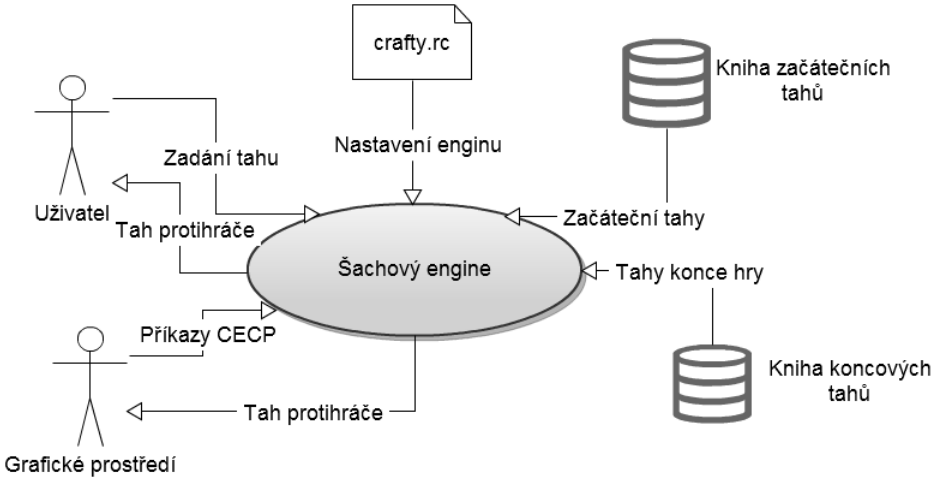
1.2 Šachový program

Termínem šachový program rozumíme nezávislý počítačový program, který vypočítává šachové pozice a tahy. V dnešní době je používání šachových programů velice oblíbené. Existuje mnoho hotových řešení, jak samotné programy, tak i celé aplikace s přívětivým uživatelským rozhraním. A v neposlední řadě se tyto programy srovnávájí pomocí šachových benchmarků nebo se pořádají soutěže v šachu proti sobě.

Tento program je schopen samostatného běhu v procesu, je navržen i pro spouštění v terminálovém okně. Ovládá se pomocí standardního vstupu a výstupu. Pro nastavení a ovládání je nutné použít jeden ze dvou nejpoužívanějších protokolů, jsou jimi protokoly UCI a CECP. Další vlastností šachového programu je vnitřní reprezentace šachového pole a kontrola správnosti tahu. Tato vlastnost je důležitá zejména pro ověření správnosti tahu či vypsání herního pole ve formě textového výpisu.

1.2.1 Program Crafty

Pro tuto aplikaci byl vybrán šachový program Crafty ve verzi 23.4. Tento program je volně dostupný program s otevřeným zdrojovým kódem. Je vyvíjen Dr. Robertem M. Hyattem, působícím na Alabamské univerzitě v Birminghamu. Distribuuje se jako zdrojový kód s přiloženým spustitelným exe souborem. [4]



Obr. 8: Systém šachového programu Crafty

1.2.2 Protokol CECP

CECP (Chess Engine Communication Protocol) je otevřený komunikační protokol zajišťující komunikaci mezi šachovým programem a grafickým prostředím. Byl vytvořen pro standardizaci komunikace mezi programem a GUI. [5] Díky tomu, že šachový program používá pro komunikaci protokol CECP, je možné ho použít s různými grafickými prostředími a naopak možnost jednoho GUI s mnoha programy. Toto je výhodné například pro testování grafických prostředí nebo možnost otestování více šachových programů či změření jejich rychlosti pomocí jedné aplikace.

Pro komunikaci využívá CECP čistě textový vstup a výstup. Jedná se tedy o jednoduché textové příkazy, většinou s parametrem, ukončené znakem pro nový řádek \n.

Použité příkazy protokolu CECP:

- sn x – nastaví maximální počet prohledávaných vrcholů na hodnotu x
- ponder on/off – volba předvídání tahu protivníka
- output – příkaz s parametrem, použitý k nastavení formátu tahu
- display – možnost zobrazení či skrytí výstupních hodnot
- end – ukončení běhu šachového programu
- move – pro provedení tahu

1.3 Formulářová aplikace

Aplikace pro zobrazení herního pole hry šachy a implementaci šachového programu je psána v programovacím jazyce C++/CLI. Tento jazyk byl vybrán kvůli kompatibilitě s knihovnou NXT++ psanou v C++. Její struktura má vzhled formulářového okna, které slouží pro prezentaci hry šachy a vnitřní struktura je naprogramována pro snadnou implementaci šachového programu a knihovny NXT++.

1.3.1 Jazyk C++/CLI

Jazyk C++/CLI je rozšířením jazyka C++ o knihovny z platformy .NET. Jedná se například o tyto knihovny:

- System::Windows::Forms
- System::Drawing

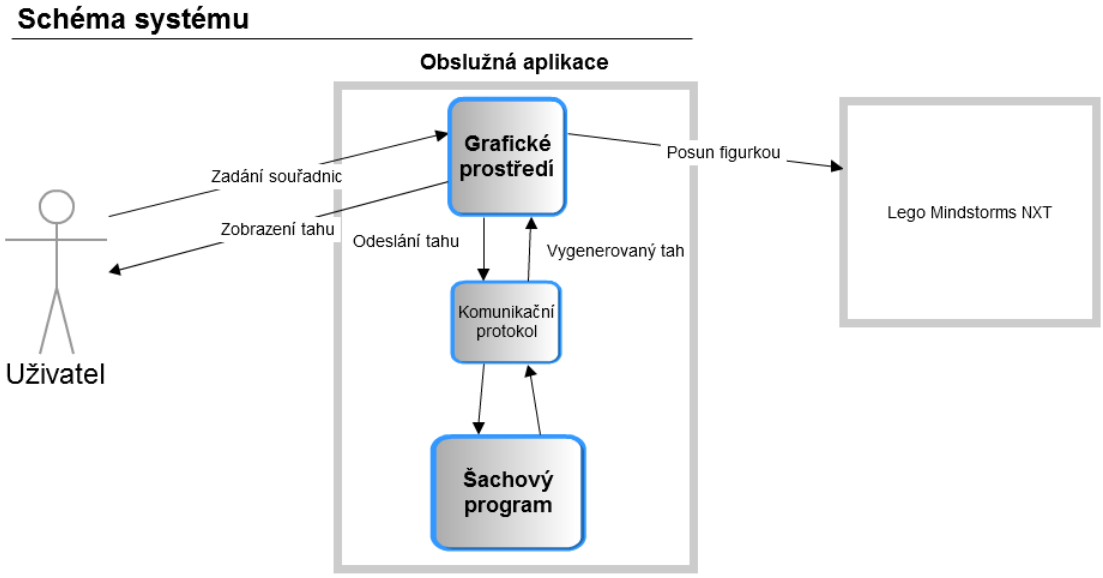
Pomocí těchto knihoven je tvoření formulářů výrazně jednodušší a například vykreslení hracího pole je pomocí knihovny Drawing velice snadné. Tento jazyk byl vybrán na základě požadavku realizace prostředí pomocí formulářové aplikace. Pomocí vhodného nastavení je ho možné kombinovat s jazykem C++. Největším rozdílem je použití objektů typu Managed Class, které jsou vytvářeny pomocí příkazu gcnew, který zajišťuje přidělení ke Garbage Collectoru z platformy .NET. Bohužel nelze kombinovat objekty typu Managed a Unmanaged, proto je nutné v této aplikaci použít oddělení pomocí pointerů, které zajistí, že tyto typy nebudou kombinovány. [3]

Objekty typu Unmanaged se vyznačují tím, že nejsou spravovány pomocí garbage collectoru, ale pomocí programátora, který v kódu zajistí, že je jejich paměť po skončení platnosti uvolněna. Tyto objekty jsou základním typem jazyka C++. Jejich vytvoření je pomocí příkazu new nebo pomocí ukazatele na daný datový typ. Tento typ objektu je použit v používané knihovně NXT++, proto je nutné vytvořit třídu a vhodně zkombinovat Managed a Unmanaged objekty.

Objekty typu Managed jsou od vytvoření spravovány pomocí garbage collectoru, který se stará o správu paměti alokované tímto objektem a pokud již na něj neexistuje odkaz, jeho paměť je uvolněna a objekt je zničen. Tento způsob správy paměti se dnes velmi často používá u jazyků, které se spouštějí přes programový framework. Jedná se hlavně o jazyk C# a Java.

2. Návrh systému automatu

Automat pro hru šachy by měl být schopný vést šachovou partii proti lidskému protějšku. Mezi jeho hlavní funkce patří vyobrazení postupu hry na obrazovce počítače, kontrola tahu uživatele s následným generováním tahu a posouvání figurkami na herní desce.



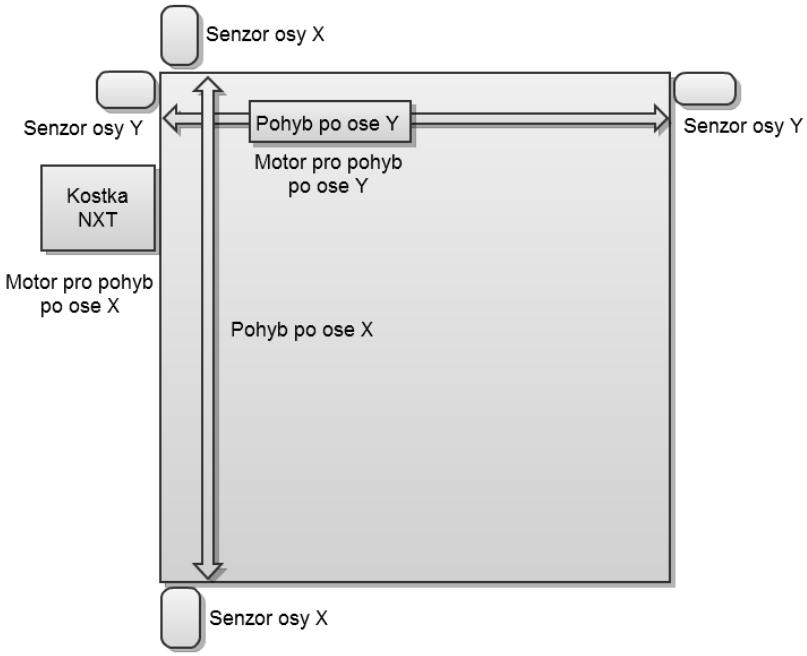
Obr. 9: Návrh systému automatu

2.1 Konstrukce Lego Mindstorms NXT

Na konstrukci automatu jsou v zadání bakalářské práce kladeny nároky na dostatečně robustní konstrukci a použití kartézského souřadného systému dvou os pro pohyb. Tato podmínka byla zásadní pro stavbu celé konstrukce.

Základem pro konstrukci je čtvercová podstava, na které jsou pomocí Lega vytvořeny dvě kolejnice pro pohyb motoru po ose x. Pomocí tohoto motoru je pohybováno s podélnou konstrukcí pro pohyb po ose y. Konstrukce pro pohyb po ose y obsahuje servomotor a kolejnice osazené ozubenými bloky. K servomotoru je připojen elektromagnet pro uchopení figurky, spojený nepohyblivě s konstrukcí servomotoru. V levé části automatu je na pohyblivé části osy y připevněna inteligentní kostka NXT. Toto umístění kostky je velice výhodné pro pozdější organizaci kabeláže. Právě kabeláž je velkým problémem pro celou funkčnost automatu. Množství kabelů vedoucí k aktivním součástkám velice často překáželo v pohybu po obou osách. Proto bylo

nutné tyto kabely vhodně umístit a připevnit ke konstrukci, aby při pohybu nebyla jakkoli omezena hybnost celé soustavy. Tato konstrukce je dále zpevněna a vyztužena pro větší stabilitu a přesnost posunu po obou osách pomocí součástek Lego Technics. Na každé ose je připevněna dvojice dotykových senzorů pro měření celkové rozteče a kalibraci po provedení každého tahu. Takto navržená konstrukce umožňuje pohyb po celém hracím poli, omezeným pouze čtyřmi dotykovými senzory. Pro uchopení figurky je použit dostatečně silný elektromagnet, který po přijetí pod figurku může předmět uchopit a v souřadném systému přemístit na cílové místo. Konstrukce by měla být dostatěčně robustní i při použití vysokých rychlostí otáčení servomotorů a tak by měl být pohyb po hracím poli dostatečně svižný.



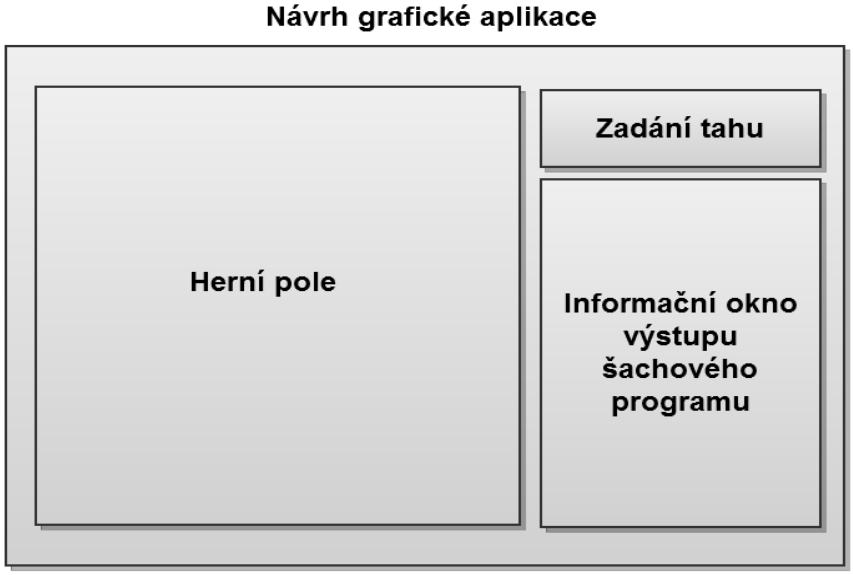
Obr. 10: Schéma pohybu po osách

2.2 Obslužná aplikace

Tato část systému obsahuje prostředky pro zobrazení grafického prostředí. To je složeno z herního pole hry šachy, nastavení šachového programu a také z informačního okna, které bude obsahovat data generovaná šachovým programem. Další částí je datová struktura pro implementaci programu pro kontrolu a hledání tahu. Ta by měla obshaovat zmíněný program, strukturu pro komunikaci s ním a metody pro zpracování dat. Poslední částí je datová struktura pro ovládání systému NXT. Pro to je použita knihovna NXT++ a vytvořena struktura tříd pro komunikaci s NXT.

2.2.1 Grafické prostředí

Pro tuto část byla zvolena formulářová aplikace psaná v jazyce C++/CLI. Tento jazyk je oproti jazyku C++ rozšířen o knihovny platformy .NET. Tyto knihovny byly využity pro vytvoření textových polí a tlačítek pro snadné ovládání aplikace a dále pro samotné vykreslení hracího pole hry šachy. Herní plán s figurkami je vykreslen na základě vlastního návrhu struktury pro hru šachy v levé části okna. Tato struktura využívá objektového programování a zejména dědičnosti, která je využita u tříd s figurkami. Pole hry je reprezentováno pomocí pole objektů abstraktní třídy figurka. Takto navržená struktura je výhodná z hlediska implementace stejného kódu všech figurek už v abstraktní třídě. V pravé horní části se nachází vstupní pole pro zadání tahu a pod ním je informační okno s daty generovanými šachovým programem.

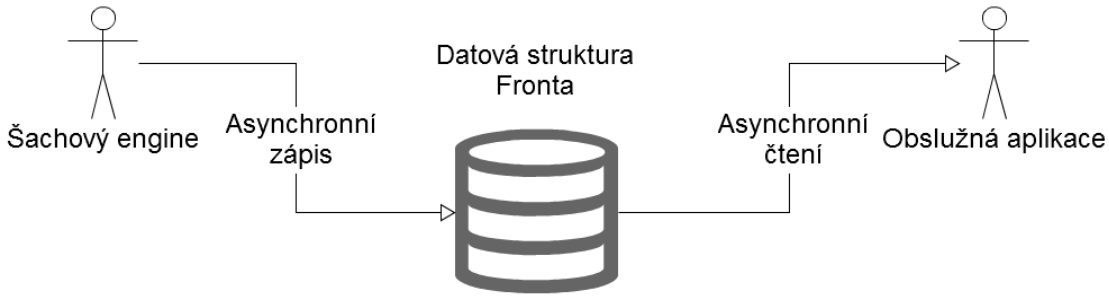


Obr. 11: Návrh grafické aplikace

2.2.2 Šachový program

Pro správnou funkčnost systému a pro sofistikované hledání tahu protihráče bylo nutné obsáhnout v aplikaci také šachový program. Ten má za úkol ověřovat správnost tahu zadaného uživatelem pomocí vnitřní reprezentace herního pole. A hlavně generovat tah protihráče. Dále bylo nutné, aby měl program otevřený a volně dostupný zdrojový kód. Pro snadné použití je vhodná také možnost nastavení formátu výstupu, maximální dobu hledání tahu a použití protokolu CECP. Všechny tyto parametry má program Crafty, který je zdarma a lze ho stáhnout ze stránek projektu. Je navržen pro spuštění

v samostatném procesu, proto byla vytvořena struktura obsahující proces a proudové operátory pro komunikaci s ním. Další funkčnost je zajištěna pomocí struktury pro přenos dat, je jí vlastní struktura s třídou fronta, která umožňuje asynchronní zápis a čtení pomocí více vláken. Té je využito při čtení dat z programu, která jsou vkládána čtecím vláknem do fronty a hlavní vlákno pak provádí operace nad těmito daty.



Obr. 12: Návrh komunikace

2.2.3 Ovládání NXT

Systém NXT je s aplikací propojen kabelem USB a pro komunikaci je použita knihovna Fantom, která obsahuje nástroje pro ovládání NXT. V aplikace je vytvořena třída pro vytvoření metod, umožňujících ovládát systém NXT. Mezi důležité metody paří například kalibrace, změření vzdáleností os, posun figurkou a vyhození figurky.

3. Obslužná aplikace

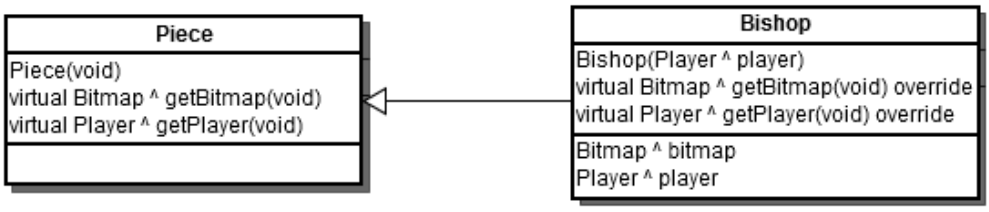
3.1 Grafická aplikace pro hru šachy

3.1.1 Grafické prostředí pro hru šachy

Herní pole se skládá ze tří částí. Nejdříve je vykreslen červený čtverec, který je překryt herním plánem s šachovnicí a vytváří tak opticky rám herního plánu. Následně jsou vykresleny všechny figurky, které používají vlastnosti průhledného pozadí a tak je možné každou figurku vykreslit jak na černé, tak bílé pole, aniž bychom museli použít dva různé obrázky figurky.

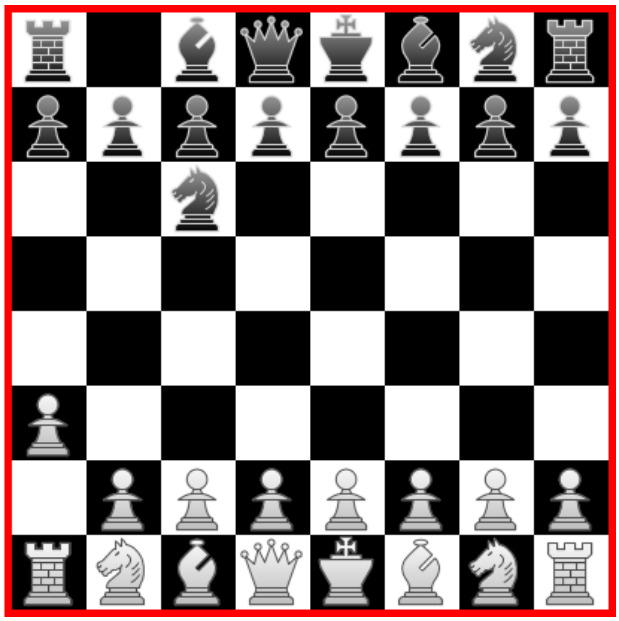
Herní plán se skládá ze 64 polí šachovnice, která mají střídavě černou a bílou barvu. Pro dobré oddělení od okolí je pod toto hrací pole přidán červený čtverec, který vytváří kolem hracího pole pomyslný rám. Pole je umístěno do panelu formulářové aplikace pro snadnější správu a přístup ke grafickým metodám pro vykreslení. Toto pole je vykresleno ihned po zapnutí aplikace pomocí metody paint, kterou vlastní hlavní okno formulářové aplikace. Aby bylo možné vykreslit herní pole je nutné nejdříve získat paletu pro kreslení na panel pomocí metody *panel.CreateGraphics* a následně již pomocí dvou cyklů vykreslit herní pole.

Pro každý typ figurky byla vytvořena třída, která obsahuje informace o hráči, který danou figurku vlastní a také bitmapu s obrázkem, který figurku znázorňuje. Pro to, abychom později mohli vytvořit pole figurek o rozměrech 8*8 polí, kde každé pole obsahuje jiný typ figurky, bylo nutné vytvořit nadřazenou třídu pro všechny typy figurek. Tato třída se jmenuje *Piece* a obsahuje metody pro získání hráče a bitmapy daného objektu.



Obr. 13: Třída reprezentující figurku

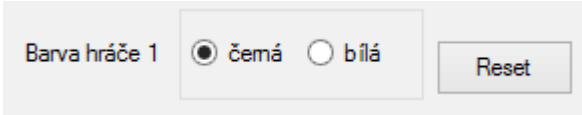
Herní pole je reprezentováno pomocí dvourozměrného pole objektů typu *Piece* a rozměry 8*8 objektů. Samotné vykreslení probíhá v metodě *Paint* hlavního formulářového okna. Toto pole je procházeno pomocí dvou cyklů for a postupně jsou na herní pole vykresleny všechny figurky. Pro každou figurku je tedy zavolána metoda *getBitmap*, která vrátí příslušnou bitmapu, která je následně vykreslena na příslušné pole herního plánu. Jednotlivé bitmapy všech figurek jsou nastaveny tak, aby byly vykresleny průhledně vůči barvě pozadí. Toto je nastaveno pomocí metody *MakeTransparent* u každé bitmapy.



Obr. 14: Herní pole

3.1.2 Ovládací prvky grafické aplikace

Na hlavním formuláři se nachází také možnost nastavení barvy protihráče a možnost resetování herního pole a šachového programu.



Obr. 15: Ovládací panel grafické aplikace

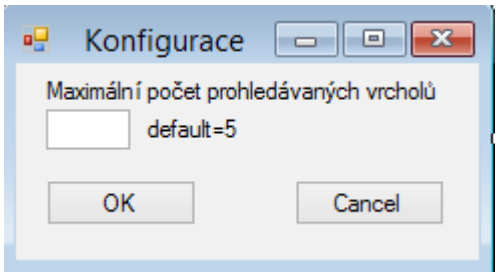
Základní nastavenou barvou pro protihráče je černá barva. Tuto barvu je možné změnit kliknutím do příslušného rádiového tlačítka v nastavení barvy. Pro změnu barvy je dále nutné potvrdit tento výběr tlačítkem *reset*, které nastaví hrací pole do základního rozestavení a provede reset šachového programu. Poté je nejprve přiřazena správná

bitmapa s obrázkem příslušné barvy k figurce. Následně je změněn vlastník figurky pomocí nastavení atributu *Player* v objektu figurky. Nakonec je zavolána metoda pro překreslení panelu s herním polem.

Pro resetování hracího pole je nutné kliknout na tlačítko reset, které nejprve resetuje rozestavení figurek. To znamená, že pole objektů s figurkami je nastaveno na základní rozestavení a barvy i vlastníci figurek nejsou změněny. Další krok je odeslání příkazu reset do šachového programu, aby byly vyresetovány i pozice figurek v programu.

3.1.3 Rozhraní pro nastavení šachového programu

Pro možnost nastavení šachového programu byla přidána do menu aplikace záložka s možností otevření okna nastavení. To je vytvořeno pomocí nové třídy rozšiřující nadřazenou třídu *Form*. Okno umožňuje nastavení programu příkazem pro omezení maximálního počtu procházených vrcholů a tím i maximální dobu hledání tahu.



Obr. 16: Konfigurace šachového programu

3.2 Implementace šachového programu

Pro implementaci a komunikaci s šachovým programem byla vytvořena třída s názvem *CECP*. Tato třída obsahuje prostředky pro vytvoření procesu pro program, komunikaci s programem a proudové operátory pro čtení a zápis do programu.

Pro vytvoření procesu pro program je zapotřebí jmenného prostoru *System::Diagnostics*, který obsahuje třídu *Process*. Pomocí této třídy je následně vytvořen a nastaven proces pro běh programu Crafty. Důležité je nastavení atributu *CreateNoWindow* na false, aby vytvářený proces nezobrazil terminálové okno. Další důležité nastavení je povolení přesměrování standardního vstupu a výstupu pomocí atributů *RedirectStandardInput* respektive *RedirectStandardOutput*. Následně je už jen

zadána cesta ke spustitelnému souboru a proces je spuštěn pomocí metody *Start* třídy *Process*.

Komunikace s šachovým programem je řešena pomocí přesměrování standardního vstupu a výstupu programu do objektů zajišťujích zpracování těchto informací. Pro zapisování dat do programu byla vybrána třída *StreamWriter*, jejíž instance je nastavena na standardní vstup procesu s programem. Takto byl přesměrován i standardní výstup do objektu *StreamReader*.

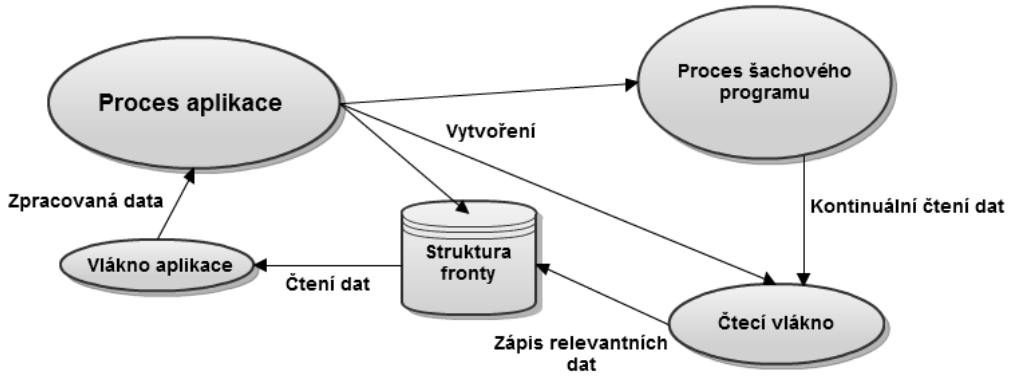
Program Crafty je navržen tak, aby byl za jakéhokoliv stavu byl schopen přijímat textový vstup ze standardního vstupu. Tato vlastnost je využita při odesílání příkazů do programu, kdy je možno kdykoliv odeslat příkaz a daný příkaz je vykonán. Textový vstup má formát jedné textové řádky ukončené znakem \n. Pro odesílání dat do programu jsou vytvořeny dvě metody. Metoda *CECP::sendMove* obsahuje kód, který odešle tah do programu a zpětně kontroluje, zda je daný tah platný. Druhá metoda je metoda *CECP::sendCommand*, která jednoduše odešle jakýkoli příkaz do programu bez zpětné vazby. Toto je využito například pro nastavení programu za běhu nebo ukončení při ukončení aplikace.

Po zadání vstupních dat do programu je vygenerován výstup na standardní výstupní proud a je možné data z programu přečíst. Tato data jsou na výstup odesílána asynchronně vůči vstupním datům. To znamená, že pokud jsou data odeslána na vstup programu, tak není zajištěno, že program do určité doby data na výstup vygeneruje. Proto bylo nutné vytvořit datovou strukturu pro asynchronní přenos dat mezi programem a obslužnou aplikací.

Nejprve byla vytvořena třída obsahující data nutná pro čtení dat z programu a následné zapsání získaných dat do struktury sdílené s obslužnou aplikací. Pro uchovávání a distribuci dat byla zvolená datová struktura fronty, zapsaná v jazyce C++/CLI jako *Queue<String* ^> ^, do které jsou data postupně vkládána na konec a odebírána z vrcholu fronty. Druhým objektem je třída pro čtení z proudu *StreamReader*, na který je přesměrován výstupní proud z programu.

Dalším krokem bylo vytvoření metody spustitelné v samostatném vlákně, která umožní nepřetržité čtení dat z programu. Tato metoda je spouštěna pomocí spuštění vlákna se vstupním parametrem, kde je předána instance na objekt třídy *CECPData*. Po

spuštění vlákna je v metodě provedena extrakce dat předaných pomocí třídy *CECPData*. Po extrakci je vytvořen cyklus while s podmínkou testující nastavení proměnné *State*, která slouží pro ukončení čtení z programu. Proměnná *State* je nastavena na false v případě ukončení aplikace, aby se vlákno pro čtení ukončilo. V tomto cyklu je dále provedeno čtení z programu pomocí příkazu objektu *StreamReader ReadLine*, který přečte jeden řádek a pokud řádek obsahuje alespoň jeden platný znak, tak je zařazen do fronty dat třídy *CECPData*.



Obr. 17: Procesy aplikace

3.2.3 Komunikační protokol CECP

Po odeslání příkazu do programu je provedeno zpracování příkazu a na výstup je vygenerován výstupní řetězec. V případě úspěšného provedení příkazu je vypsáno hlášení o nastavení nebo provedení příkazu. Pokud nastane jakákoli chyba, program vypíše o jakou chybu se jedná.

Šachový program Crafty je po zkompilování nastaven na výchozí hodnoty, z nichž je některé potřeba přenastavit pro použití v této aplikaci. Po spuštění procesu pro program je vypsána verze a nastavení programu. Prvotní nastavení je možné provést přes konfigurační soubor *crafty.rc*, který obsahuje příkazy pro nastavení programu. Pokud je nastavení úspěšně provedeno, program vypíše po spuštění dané nastavení.

Důležitá nastavení pro obslužnou aplikaci:

- ponder off – *Pondering* je vlastnost programu, která v době po nalezení tahu a čekání na uživatelský vstup hledá a předpovídá tah protihráče. Tato vlastnost byla vypnuta, protože vytěžuje procesor předvídáním tahu a pro tuto aplikaci nemá žádný význam.

- output long – Toto nastavení upravuje formát výstupu z programu po nalezení tahu. Nalezený tah je vypsán jako algebraický výraz celého tahu např. Nb1c3, kde první znak znamená typ figurky a další ctyři znaky označují úvodní a cílovou pozici tahu. Tato formulace je výhodná pro pozdější zpracování a provedení tahu.
- sn 5 – Nastavuje maximální počet prohledávaných vrcholů při hledání tahu. Tato možnost nastavení je uvedena v manuálu protokolu CECP jako možnost nastavení úrovně obtížnosti programu nebo také možnost omezení doby vyhledávání. Tento parametr je možné kdykoliv v průběhu hry změnit a tím zvýšit či snížit obtížnost a dobu hledání tahu. Pro tuto aplikaci je zvoleno nastavení sn 5, což je velmi malé číslo a tím pádem je tah nalezen skoro okamžitě, avšak není z hlediska obtížnosti ten nejlepší.

Komunikace s CECP probíhá ve čtyřech krocích:

1. Po spuštění procesu s programem je provedeno úvodní nastavení pomocí souboru *crafty.rc*, který obsahuje příkazy pro nastavení programu této aplikace.
2. Po úvodním nastavení programu ohlásí připravenost pomocí výzvy pro zadání tahu bílého hráče výstupem: White(1):, kde white znamená barvu hráče a číslo v závorce označení tahu. V tento moment je program připraven přijímat příkazy pro další nastavení nebo samotný tah hráče.
3. Po zadání příkazu do programu je vygenerován výstup na základě vstupních dat. Pokud program přijme jako vstupní data tah protihráče, je ihned generována cesta hledání tahu. Program prochází jednotlivé vrcholy svého grafu a vypisuje průběh procházení. Nakonec, když je nalezen nejlepší tah, vypíše výstup ve formě Black(2): Nb1c3, kde N znamená typ figurky a algebraická notace startovní a cílové pole tahu.
4. Pokud chceme ukončit šachový program a tím i celý proces, je nutné odeslat na standardní vstup programu příkaz *end*.

3.2.4 Zpracování dat šachového programu

Jak již bylo dříve zmíněno, data z programu Crafty jsou postupně ukládána do struktury fronty, kde čekají na zpracování. Odesílání dat probíhá přes implementované

metody, které zapisují data přímo do vstupního proudu programu.

Pro zpracování a následné předání dat do programu jsou v aplikaci vytvořeny metody pro jednotlivé příkazy.

1. Zápis libovolného příkazu do programu Crafty obsluhuje metoda *CECP::sendCommand*. Tato metoda nemá žádný návratový parametr a není tedy možné ověřit správnost provedení daného příkazu přímo v kódu. Metoda je používána hlavně pro nastavení programu, kdy je odeslán daný příkaz do programu a správnost provedení musí ověřit uživatel v informačním okně formulářové aplikace.
2. Zápis šachového tahu obsluhuje metoda *CECP::sendMove*, která má jako návratovou hodnotu řetězec typu *String*, ze které je možné zjistit, zda se daný tah provedl v pořádku. Tah je do metody předán v algebraickém formátu xXyY, kde x a X znamená písmeno a číslo začátečního pole tahu a y a Y cílové pole tahu. Tento výraz je poslán do programu a následně je zkontrolována správnost daného tahu. Pokud je tah neplatný, program vygeneruje na výstup text „Illegal move“. Správnost tahu je kontrolována vždy po každém tahu prohledáním výstupu z programu, tedy celé fronty výstupu, zda neobsahuje signalizační řetězec. V případě chybného tahu je jako návratová hodnota předán řetězec „Illegal move“, v opačném případě je úspěch signalizován řetězcem „úspěch“.
3. Pro ukončení spojení s programem je vytvořena metoda *CECP::close*. Tato metoda obsahuje pouze dva příkazy. První z nich je příkaz nastavující příznak pokračování čtecího vlákna ve čtení dat z programu. Po zavolání této metody je ukončeno pokračování pomocí nastavení statusu čtecího vlákna na false. Toto vlákno dále čeká na poslední výstup z programu. Ten je vygenerován pomocí příkazu end, odeslaného na vstup programu. Tento příkaz ukončí běh programu, je vygenerován poslední výstup z programu, který ukončí čtecí vlákno. A nakonec je samovolně ukončen i proces programu.

Výstupní data z programu jsou uložena ve frontě, ze které jsou později všechna zobrazena v informačním okně formulářové aplikace. Tato data je nutné dále zpracovat. Část zpracování dat je provedena již při vkládání do fronty a to kontrola, zda obsahují platné znaky. Pro získání tahu generovaného programem je potřeba nejprve data

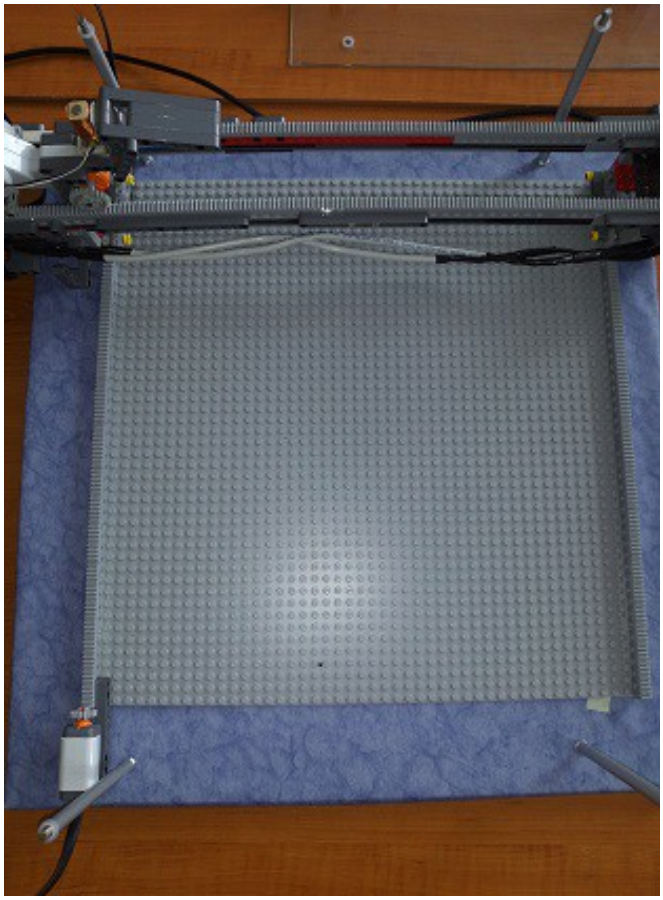
zpracovat a rozlišit na důležité záznamy, mezi které řadíme řetězce s textovým vyjádřením barvy protihráče. Pro tuto úpravu byla vytvořena metoda *CECP::getMove*, která zajistí předání algebraického výrazu tahu. Jelikož fronta obsahuje množství dat, která jsou pro získání tahu nedůležitá, je nutné nejdříve vyhledat záznam pomocí příznaku příkazu *String.contains*, který obsahuje informaci o tahu. Standardním výstupem tahu je řetězec např.: Black(1): Nb8c6. Tato metoda tedy prohledá všechny záznamy ve frontě a v případě, že nalezne hledaný záznam, který obsahuje řetězec Black, je tento záznam dále zpracován. Nejdříve je provedeno rozseknutí řetězce podle znaku mezery, kdy v nově vytvořeném poli na pozici 1 je výraz Nb8c6. Takto upravený výraz je odeslán jako výstup metody a je dále zpracováván metodami pro posunutí figurky a vykreslení figurky na správném místě.

4. Šachový automat z Lego Mindstorms NXT

Posledním krokem bakalářské práce bylo vytvoření robustní konstrukce pro hru šachy pomocí Lego Mindstorms NXT a následné propojení s obslužnou aplikací a realizování hry šachy. Pro konstrukci systému je použita podstavná dřevěná deska, která pomocí čtyř závitových tyčí drží skleněnou desku, na které jsou umístěny hrací figurky. Na dřevěné desce je dále umístěna deska Lego a na této desce je vytvořena konstrukce umožňující hru šachy.

4.1 Konstrukce systému

Základem celé konstrukce je čtvercová deska Lego o rozměrech 50x50 centimetrů, která slouží jako podstava pro celý systém. Na této desce jsou dále umístěny kostky Lega tak, aby vytvořily ozubené kolejnice, které budou dále využity pro pohyb po ose x.



Obr. 18: Herní deska

Pohyb po ose x je realizován pomocí ozubeného kola pohyblivé části systému otáčejícím se po ozubené kolejnici na podstavné čtvercové desce. Pohyb zajišťuje servomotor připevněný k pohyblivé části na jedné straně. Zde je také převeden otáčivý pohyb motoru na ozubené kolo. Aby byl zajištěn rovnoměrný pohyb obou stran, je mezi oběma stranami umístěna tyč, která zajišťuje synchronizaci ozubených kol na každé straně tím, že jsou staticky propojena. Aby byla zajištěna potřebná stabilita a systém se při pohybu nepřevrhl, jsou zmíněná ozubená kola doplněna dalšími dvěma koly na každé straně. Takto postavený systém pohybu po ose x zajišťuje stabilitu při pohybu a je možné využít vysokých rychlostí motoru.

Další součástí pohybu po ose x jsou dva dotykové senzory umístěné vždy v hraničním bodu pohybu tak, aby nedocházelo ke kolizi s vnějším prostředím a zároveň aby byl využit plný potenciál herního pole. Jeden senzor je tedy umístěn u dolního okraje podstavné desky a druhý na opačném konci. Toto je využito hlavně při měření vzdáleností mezi hraničními body. Dále je to využito při výpočtu posunu o jedno šachové pole, což je poměrná část z této vzdálenosti.

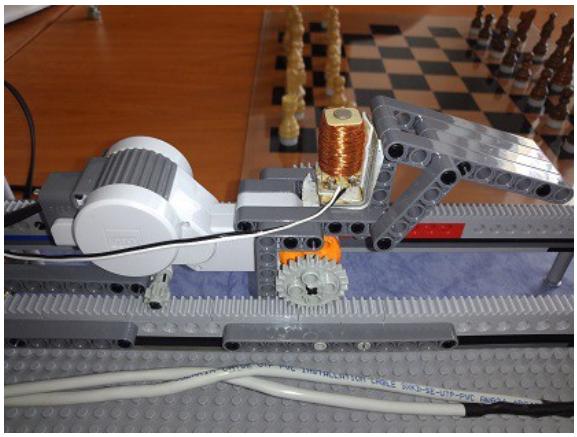
Pro pohyb po ose y je vytvořen mezi dvěma částmi osy x most. Tento most je na každé straně připevněn ke konstrukci osy x zpevňujícími díly stavebnice Lego tak, aby byla zvýšena přesnost a pevnost celé konstrukce. Dále je most vyztužen dalšími díly, aby nedocházelo k prohybu nebo deformaci mostu. Po celé délce mostu je na každém kraji připevněna kolejnice s ozubeným povrchem. Tato konstrukce umožňuje použití ozubených kol pro pohyb po ose y.

Pohyb po ose y je realizován pohyblivou částí se servomotorem a elektromotorem. Tato část je na kolejnicích připevněna pomocí spodních přítlačných dílů, které brání pohybu vzhůru. Je tím tedy zajištěn dostatečný přítlak této části a nedochází k vykolejení nebo přesmykům na ozubených částech stavebnice.

Součástí pohyblivé části je elektromagnet, který je připevněn na díly Lego pomocí oboustraně přilnavé pásky, zajišťující její fixaci. Dále jsou tyto díly nepohyblivě spojeny s elektromotorem a je možné pomocí této konstrukce při zapnutém elektromagnetu pohybovat s figurkami umístěnými nad skleněnou deskou.

Další částí pohyblivé konstrukce na ose y je plošina umožňující vyhození figurky mimo hrací pole. Automatu stačí pouze přijet pod kraj skleněné desky, tím se

figurka nahne a pomocí této plošiny sjede na připravené místo. Tato plošina je postavena pomocí dílů s úhlem náklonu 45° a figurka tedy samovolně sklouzne po plošině dolů.



Obr. 19: Servomotor s elektromagnetem

Jelikož používaná kabeláž je velmi nepoddajná a velmi často docházelo k uváznutí některého z kabelů, byl přidán na pohyblivou část osy y díl Lego, který je dostatečně dlouhý a umožňuje na sebe navázat kabely a ty poté navést tak, aby nepřekážely v pohybu.

Vzhledem k velmi tvrdé kabeláži bylo nutné vyřešit umístění kostky NXT tak, aby byla v dostatečné vzdálenosti ke všem aktivním prvkům konstrukce a nedocházelo k uvíznutí v důsledku překážení kabeláže. Po mnoha pokusech o vyřešení byla kostka umístěna přímo na pohyblivou konstrukci osy x. Vzhledem k velké váze bylo nutné patřičně vyztužit konstrukci kolem kostky NXT a zejména v místě, kde je uchycena. Dále je ještě podepřena další výztuhou ze spodní části konstrukce osy x. Takto umístěná kostka umožňuje použít výhodně kabely pro propojení všech částí systému.

Obr. 20: Šachový automat NXT

4.2 Propojení a implementace NXT v C++/CLI

Pro ovládání systému NXT pomocí programovacího jazyka C++/CLI je nutné implementovat ovládácí knihovnu NXT++. Tato knihovna obsahuje potřebné nástroje pro ovládání systému a skládá se ze tří částí. První částí jsou hlavičkové soubory NXT++, které je nutné přidat do projektu s obslužnou aplikací. Jedná se o soubory NXT++.h, comm.h a resource.h obsahující nutné jmenné prostory a deklarace proměnných pro správnou funkčnost knihovny NXT++. Další částí jsou zdrojové soubory C++, které obsahují kódy všech tříd knihovny. Poslední částí je kompilovaná knihovna Fantom, kterou používá systém NXT++ jako nástroj pro provedení jednotlivých operací. Všechny tyto soubory je nutné zahrnout do kompilace aplikace a důležitou součástí je také nastavení kompilátoru C++/CLI. Ten je ve výchozím nastavení nastaven pro kompilaci čistě kódu C++/CLI. Pro tuto aplikaci bylo nutné zvolit možnost kompilace C++/CLI s podporou C++, jelikož knihovna NXT++ je psána v jazyce C++.

Aby bylo možné ovládat systém NXT pomocí vlastních metod přímo z kódu, byla vytvořena třída *NXTDevice*, která obsahuje vlastní metody pro snadné a intuitivní ovládání. V této třídě byly postupně vytvořeny metody pro inicializaci systému NXT, kalibraci systému souřadnic, posunutí figurky, vyhození figurky a posunu do základního postavení.

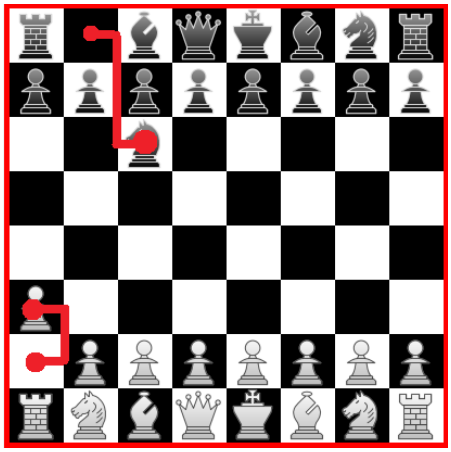
- *NXT::Inicializace* – Tato metoda slouží k otevření spojení s kostkou NXT, které je otevřeno pomocí příkazu knihovny *NXT::OpenNXTDevice*, jako parametr se předává název kostky a zda je kostka připojena pomocí rozhraní USB nebo Bluetooth. Pokud se podaří otevřít spojení, metoda předá návratovou hodnotu True a následuje nastavení senzorů. Senzory připojené ke kostce jsou všechny dotykového typu a je nutné je nastavit pomocí příkazu *NXT::Sensor::SetTouch*. Nastavení je provedeno pro senzor na každém portu. Nutnost tohoto nastavení spočívá v nastavení rozsahu výstupních hodnot. Pro dotykový senzor požadujeme pouze hodnoty logická 1 a 0. V této metodě je ještě nastavena rychlost posuvu servomotorů obou os při posuvu s figurkou a bez figurky. Hodnota posuvu osy x s figurkou je 50%, bez figurky 80% maximální rychlosti servomotoru. Posuv na ose y je nastaven na 25% při posuvu s figurkou a 40%

bez figurky. Rozdílné nastavení je způsobeno velikostí ozubených kol na osách.

- *NXT::zmerVzdalenostiOs* – Pro změření maximální vzdálenosti na ose je použito dvou dotykových senzorů a senzoru otáček v servomotoru.
  - Nejdříve je nastaven dopředný chod na servomotoru a konstrukce se pohybuje k senzoru, dokud není sepnut. Při sepnutí senzoru je motor zastaven, je vynulován senzor otáček a následně je nastaven zpětný chod motoru dokud konstrukce nedorazí ke druhému senzoru. Jakmile se sepne druhý senzor, je motor zastaven a je uložena hodnota otáček ze senzoru motoru. Tato hodnota se nadále používá pro určení vzdálenosti posunu o jedno políčko. Tímto způsobem jsou vzdálenosti na obou osách a výsledkem je tedy zjištěná vzdálenost posunu o jedno pole.
- *NXT::goToStart* – Po každém tahu figurkou nebo vyhození figurky je nutné vrátit se do základní pozice. Tuto funkci zajišťuje metoda *goToStart*, která po zavolání nastaví motory na obou osách do zpětného chodu a takto se otáčejí až do sepnutí příslušných senzorů.

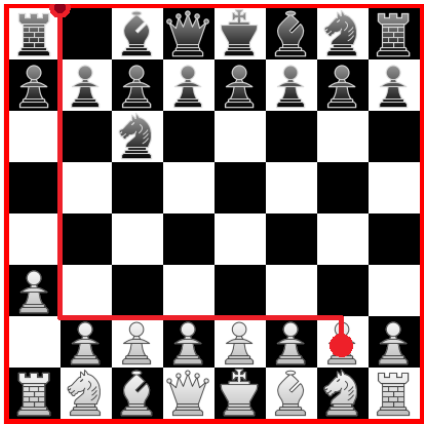
Posunutí figurky obstarává metoda *NXT::posunFigurkou*, která přejímá parametry posunu zdrojové souřadnice tahu na ose x a y a cílové souřadnice x a y. V průběhu metody je pak proveden posun figurky pomocí vlastního algoritmu ze zdrojové pozice do cílové. Výsledný algoritmus je popsán níže.

1. Nejprve je nutné posunout automat pod figurku, kterou je taženo. Je tedy vynulován senzor otáček na servomotoru a pomocí metody *NXT::Motor::GoTo* je motor otočen o přesný počet otáček vypočtený ze vztahu: (pozice na ose x) * (dílčí posun o jedno pole na ose x). Tento postup je proveden pro obě osy. Následně je zapnut elektromagnet, který uchopí figurku nad ním.
2. Dále je vypočten posun o půl pole na obou osách, který je ve směru pohybu s figurkou, takže nedochází k prodloužení trasy posunu. V tuto chvíli se figurka nachází na mřížce šachového pole a je možné s figurkou táhnout na cílové pole aniž by došlo ke kolizi s ostatními figurkami.
3. Po posunu figurky na cílové pole je nutné vrátit zpět figurku na střed pole, toho je dosaženo pomocí vrácení posunů o půl pole.
4. Posledním příkazem je uvolnění elektromagnetu a přesun do základní pozice.



Obr. 21: Znázornění posunu

Vyhození figurky je navrženo jako posun figurky na okraj skleněné desky, na které jsou umístěny figurky. Nejprve je automat přesunut pod figurky, jsou provedeny posuny o půl pole tak, aby nedošlo ke kolizi při posunu na vyhazovací místo. Poté je figurka přesunuta na okraj skleněné desky, o který se nakloní ve směru od hracího pole a po uvolnění elektromagnetu a odjetí automatu je pomocí vyhazovací plošiny vyhozena do přistaveného boxu.



Obr. 22: Znázornění vyhození

Závěr

V této bakalářské práci bylo provedeno sestavení několika jednoduchých konstrukcí a seznámení se s jednotlivými díly stavebnice. Následně bylo vyzkoušeno programování inteligentní kostky NXT přímo z prostředí v kostce a také programování kostky přes USB z prostředí Microsoft Visual C++ pomocí knihoven pro ovládání NXT. Pro programování byly vyzkoušeny knihovny pro jazyk C++, přičemž vybrána byla knihovna NXT++ vzhledem ke své obsáhlosti a jednoduchosti používání.

Pro ovládání byla vytvořena formulářová obslužná aplikace s grafickým prostředím umožňujícím realizaci hry šachy. Aplikace je napsána v programovacím jazyce C++/CLI a používá standradní knihovny z platformy .NET. Bylo vytvořeno herní pole o velikosti 8*8 polí a vykreslení figurek probíhá pomocí průhledného pozadí. Součástí aplikace je také informační okno šachového programu a formulář pro nastavení šachového programu.

Dále bylo nutné vybrat šachový program s vlastní reprezentací šachového pole a kontrolou správnosti tahu. Byly vyzkoušeny šachové programy Stockfish, Critter, Houdini a Crafty, z nichž byl vybrán Crafty. Pro komunikaci mezi obslužnou aplikací a programem byl použit protokol CECP. Byl vytvořen proces pro běh programu a komunikační rozhraní využívající struktury fronty a používající vlákna k paralelnímu čtení dat z programu. Program Crafty byl dále nastaven pomocí protokolu CECP na požadované hodnoty výstupního řetězce a doby vyhledávání tahu.

Nakonec byl vytvořen šachový automat ze stavebnice Lego Mindstorms NXT, využívající souřadného systému dvou os. Byla vytvořena konstrukce pohybující se po ose x pomocí servomotoru a ozubených kol, která nese servomotor pro pohyb po ose y a elektromagnet pro uchopení a posun figurky. Pohybuje se ve dvou rychlostech, pro pohyb bez figurky je nastavena vyšší rychlost než pro pohyb s figurkou.

Takto navržený automat a obslužná aplikace jsou schopny realizovat šachovou partii uživatele proti šachovému programu. Po tahu každého hráče jsou figurky posunuty na místo tahu a v grafickém prostředí aplikace je tentýž tah zobrazen na displeji.

Použitá literatura

[1] FLOYD KELLY, James. *LEGO MINDSTORMS NXT 2.0: The King's Treasure*. 2010. vyd. United States of America: Apress, 2009. ISBN 978-1-4302-2491-4.

[2] GASPERI, Michael a Philippe HURBAIN. *Extreme NXT: Extending the LEGO MINDSTORMS NXT to the Next Level*. United States of America: Apress, 2009. ISBN 978-1-4302-2453-2.

[3] HEEGE, Marcus. *Expert C++/CLI: .NET for Visual C++ Programmers*. United States of America: Apress, 2007. ISBN 978-1-59059-756-9.

[4] Crafty Chess page. HYATT, Robert M. *Crafty Chess page* [online]. 2012 [cit. 2013-05-13]. Dostupné z: www.craftychess.com

[5] Protocol CECP. *Chess Engine Communication Protocol* [online]. [cit. 2013-05-13]. Dostupné z: http://www.gnu.org/software/xboard/engine-intf.html

Obsah přiloženého CD

- Text bakalářské práce ve formátu PDF
- Projekt vytvořený v Microsoft Visual Studio 2012 obsahující kód obslužné aplikace
- Spustilený soubor s šachovým programem