České vysoké učení technické v Praze
Fakulta elektrotechnická

K09 Lab Propagator

Bakalářská práce

Vedoucí práce:
Ing. Pavel Růžička

Ondřej Nývlt

Praha 2006

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne ……. …….

podpis

Annotation
Nývlt, O. K09 Lab Propagator.
Bachelor thesis. Praha, 2006.
In this text there is resolved using of PDA Asus MyPal A716 like a remote controller of presentation in PowerPoint and of model (automat) "Color Sorter" which is placed in the room K09 od DCE. In this thesis, there is described concrete solution of this questions.

Anotace
Nývlt, O. K09 Lab Propagator.
Bakalářská práce. Praha, 2006.
Práce se zabývá možnostmi využití PDA Asus MyPal A716 jako bezdrátového dálkového ovladače pro ovládání prezentace promítané aplikací PowerPoint a pro prezentaci modelu (automatu) "Třídění kuliček" umístěného v místnosti K09 katedry řízení. V práci je popsáno konkrétní řešení těchto otázek.

Poděkování:

Na tomto místě bych chtěl poděkovat Ing. Ondřejovi Dolejšovi, bez jehož vydatné a ochotné pomoci, by tato práce nikdy nemohla být dokončena.

Obsah

1 Úvod a cíl práce

Cílem práce je realizovat dva de facto nezávislé projekty spojené tím, že oba budou na dálku řízeny školním PDA. Prvním projektem je návrh a realizace dálkového ovládání prezentace spuštěné a promítané přes aplikaci Microsoft Power Point. Tento projekt se tedy skládá z programu pro PDA a PC. Základním předpokladem je, že se prezentace v aplikaci Power Point dá ovládat z klávesnice. První částí je aplikace pro PDA, která po virtuální sériové lince vytvořené pomocí BlueTooth technologie posílá při zmáčknutí HW či SW tlačítka na PDA znaky do servisní aplikace na PC, která je interpretuje jako posílání událostí zmáčknutí klávesy aplikaci Power Point, která příchozí událost bere jako skutečný vstup z klávesnice (šipka vpravo -> přejdi na další slide atd.). Výhodou je, že uživatel nemusí pozorovat displej PDA, aby mohl prezentaci ovládat, ale stačí, aby si pamatoval, které HW tlačítko PDA kterou operaci při zmáčknutí vyvolá.

Druhým projektem je návrh a realizace dálkového ovládání vybraného modelu z laboratoře K09 opět pomocí PDA. Řešení se opět skládá ze dvou částí - programu pro PDA a programu pro PLC (programovatelný automat, jenž řídí model). Program pro automat WAGO je složen z několika podprogramů – pro komunikaci, algoritmus automatického třídění kuliček a manuální ovládání modelu. Chod automatu je ovládán z programu v PDA opět přes Bluetooth virtuální sériovou linku. V tomto programu uživatel vybere, jaký mód chce spustit, zda automatický či manuální. V manuálním módu dál ovládá jednotlivé akční členy přes displej PDA. V automatickém módu pak jen spouští a ukončuje chod algoritmu třídění.

Jednotlivá řešení jsou popsána a detailně rozebrána v následujících kapitolách. Nejdříve se vždy budu zabývat vývojem svých návrhů na řešení konkrétního problému, poté detailně popíšu vybrané a realizované řešení (včetně rozebrání zdrojových kódů). V další části jsou vysvětleny a popsány problémy, které se mého řešení týkají, a pak následuje velmi důležitá kapitola „Manuál koncového uživatele“, která krok po kroku říká ‚jak má uživatel software i hardware ovládat.

Celkově se dá říci, že cílem je vytvořit pro laboratoř K09 a katedru řízení programové vybavení, které efektně i efektivně a zajímavým moderním způsobem umožní prezentovat její výsledky a projekty.

1.1 ASUS MyPal A716

Pro řešení zadání mi bylo katedrou řízení zapůjčeno PDA neboli kapesní počítač značky ASUS, neboť bez možnosti vývoje přímo na cílovém stroji by se úlohy špatně řešily. Specifikace PDA:

- Procesor: Intel 400MHz PXA255
- Operační systém: Microsoft Windows Mobile 2003 Premium edition (WinCE 4.2)
- Paměť: RAM 64 MB :: SDRAM
  ROM 64 MB :: FlashROM :: přepisovatelná :: 25 MB pro bezpečné uložení dat
- Displej: 3.5" TFT, 240 x 320, 65536 barev
- Sloty: CompactFlash type II, Secure Digial / MultiMedia Card
- Komunikace: IRDA (FIR @ 4Mbps), WLAN 802.11b, Bluetooth 1.1
- Baterie: Li-Ion 1500mAh, až 19 hodin provozu
- Rozměry a hmotnost: 135 x 78 x 17,6 mm, 197 g

Obr 1.1: Asus MyPal A716

Zařízení je vybaveno operačním systémem WM 2003 PE, tudíž jsem mohl k vývoji softwaru využít své znalosti jazyka C# a vývojového prostředí Microsoft Visual Studio, které přímo vývoj programů pro PDA s OS Windows Mobile / PocketPC nabízí.

Pro řešení zadání je dále velmi důležité, jakými bezdrátovými technologie je PDA vybaveno. U ASUS MyPal A716 jsou to infračervený port, Bluetooth a WiFi (802.11b). Řešit zadání pomocí infračerveného portu jsem vyloučil, protože má příliš malý dosah, infra porty na sebe musí být při komunikaci namířeny a signál nepronikne skrz překážky. Komunikaci pomocí WiFi jsem taky nezvolil, protože programování WiFi je obtížné a žádný z fungujících modelů v K09 není vybaven WiFi modulem. Takže nakonec zbyla realizace komunikace prostřednictvím Bluetooth. Výhodou je, že Bluetooth modul v PDA obsahuje SPP profil, což znamená, že lze realizovat jednoduše komunikaci pomocí virtuální sériové linky (tuto linku používám v řešení obou úkolů). S touto virtuální sériovou linkou se pak pracuje jako s normální "hardwarovou" linkou a sériovými porty. Pro práci se sériovými porty v PDA jsem využil knihovnu OpenNET Smart Device Framework. Další výhodou použití Bluetooth technologie je možnost jednoduchého dovybavení modelů řízených PLC WAGO Bluetooth adaptérem, který se připojí do slotu pro servisní sériový kabel. Pro připojení k PC a ovládání stačí jakýkoli z dnes prodávaných levných BT adaptéru, protože všechny dnes obsahují SPP profil. Další výhodou, především oproti IR, je daleko větší dosah - zvláště pak v laboratoři K09, kde je málo překážek (odzkoušený dosah signálu od PDA k modelu třídění kuliček je několik metrů).

2 Dálkové ovládání prezentace v Power Pointu

2.1 Úvod

Tento problém jsem řešil jako první, protože se zdál být jednodušší než ovládání modelu v laboratoři K09. Prapůvodní myšlenkou, jak ovládání vytvořit, byla idea, že se PDA připojí k PC jako zařízení HID (Human Interface Device Profile). HID je profil, který specifikuje použití zařízení

s Bluetooth čipem, která jsou ovládána člověkem. Tedy jinak řečeno např. propojení bezdrátové myši nebo klávesnice s počítačem nebo popř. mobilním zařízením. Zařízení, která jsou vybavena tímto profilem, lze tak bez nějakého zvláštního přídavného software (již součástí ovladače, ovšem starší BT (Bluetooth) moduly tento profil obvykle neobsahují - dnes ho má každý nový BlueTooth modul pro PC) použít pro ovládání PC - nejedná se jen o BT myši a klávesnice, ale i o mobilní telefony (viz třeba Sony Ericsson, modely K700 a S700). Pomocí mobilního telefonu lze tak pomocí aplikací v telefonu simulovat práci myši, ovládat Media Player, či mnoho jiných aplikací...(WinAmp etc.) Tvorba potřebného programového vybavení pro mobilní telefony je velmi jednoduchá. Mimo jiné tyto telefony dokáží ovládat právě i Power Point a promítání prezentace.

Naším předpokladem tedy bylo, že i PocketPC obsahuje HID profil a bude tak velmi jednoduché a elegantní ovládat Power Point. Ovšem z této iluze jsem byl po krátkém průzkumu a rozhovoru se zkušenými programátory vyveden. Proto se muselo hledat jiné řešení, které ovšem již nebude tak elegantní a jednoduché.

Druhým nápadem bylo nalézt komponentu v API přímo od Microsoftu, která by umožňovala se k Power Pointu připojit a ovládat ho. Takovou komponentu jsem bohužel nenašel. Jediné, co se nabízelo, bylo vytvořit si vlastní Power Point aplikaci, což ovšem není jednoduché ani potřebné.

Finálním nápadem bylo simulovat klávesnici a posílat Power Pointu "Key Strokes", tedy událost zmáčknutí příslušné klávesy a hodnotu klávesy (šipka vpravo, Enter , F5...). Tato idea se stala rozhodující pro výslednou realizaci. Toto řešení tedy ale vyžaduje dvě samostatné části programového vybavení: část pro desktop (tedy pro MS Windows), která bude přijímat přes Bluetooth z PDA příkazy a interpretovat je jako posílání zprávy aplikaci Power Point, druhá část je pak software pro PDA (v našem případě Asus MyPal A716), který po zmáčknutí daného onscreen tlačítka či hardbutton (tedy hardware tlačítka) vyšle příkaz na desktop. Obě části jsou propojeny pomocí Bluetooth virtuální sériové linky - tedy jako bychom komunikovali přes normální sériový port (jednoduchá realizace komunikace a adresace portu). To znamená, že PDA pošle přes sériovou linku znak a desktop ho přes COM port přijme, zpracuje a vyšle zprávu Power Pointu. Nakonec jsem ještě přidal ovládání též pomocí hardwarových tlačítek PDA, takže uživatel nemusí vůbec pozorovat displej PDA.

2.2 Realizace

Pro tvorbu softwaru (desktopové i PDA části) jsem zvolil programovací jazyk C# z .Net. Vývojové prostředí pro PDA část jsem zvolil MS Visual Studio 2003 a pro desktopovou část MS Visual Studio 2005.

2.2.1 Desktopová část

Desktopová část řešení představuje servisní aplikaci, napsanou v C#, která interpretuje data přijatá po virtuální sériové lince jako zprávy, jež posílá vybrané aplikaci jako stisky kláves. Tedy vlastně simuluje klávesnici. Celá aplikace je velmi jednoduchá - nejjednodušší jak jen to jde, ale zároveň co nejbezpečnější. Snažím se co nejvíce chyb, které může uživatel vytvořit, zachytit a upozornit na ně. Ve zkratce lze shrnout celé fungování takto - uživatel vybere prezentaci, zadá číslo portu, na kterém běží Bluetooth virtuální sériová linka (SPP), poté se spustí odposlech na tomto portu a spustí se Power Point s vybranou prezentací. Při příchodu dat po sériové lince se přečte jeden znak a vyhodnotí se. Pokud je k němu zaregistrována funkce (tedy stisk klávesy), pak se nalezne okno s daným jménem, aktivuje se (tedy přesune se do popředí - do „foreground“) - tyto dvě funkce jsou importovány z DLL - a pomocí funkce SendKeys.SendWait() se pošle zpráva obsluhující simulovaný stisk klávesy (dle značení od Microsoft) oknu, které je aktivní (v popředí).
Podívejme se teď podrobně na zdrojový kód:

- Co se týče použitých Namespace, tak kromě obvyklých System.Windows.Forms atd. nutných pro GUI, jsou zde dvě zásadní Namespace a to: System.IO.Ports pro práci se sériovou komunikací a System.Runtime.InteropServices, kde se nachází zásadní funkce SendKeys.SendWait, která posílá simulované stisky danému oknu, dále práce s procesy:

```
using System.IO.Ports;
using System.Runtime.InteropServices;
```

- Pro posílání stisknutí klávesy jsou důležité dvě funkce *FindWindow()* a *SetForegroundWindow*, které je nutno naimporotvat ze standartní knihovny user32.dll. :

```
[DllImport("user32.dll")]
public static extern int FindWindow(
string lpClassName,
string lpWindowName
);

[DllImport("user32.dll")]
public static extern int SetForegroundWindow(int hwnd);
```

- Konstruktor objektu Form1 (hlavní obrazovky, tedy de facto celé aplikace). Zavolá se inicializace GUI, pak zablokuji a odblokuji příslušná tlačítka (aby šlo nastavit parametry, ale bez vybrání prezentace nešel spustit odposlech na portu). Dále se nastaví parametry průzkumníka (OpenFileDialog) pro výběr prezentace - úvodní adresář, typ souborů, které se budou zobrazovat, a ukládání posledního navštíveného adresáře. Inicializace GUI je z automaticky generovaného kódu Visual Studia 2005. "Vybrat" je speciální komponenta OpenFileDialog pro výběr souborů.

```
public Form1()
{
   InitializeComponent();
   vyber.Enabled = true;
   portBlueTooth.Enabled = true;
   ukoncitPoslech.Enabled = false;
   start.Enabled = true;
   vybrat.InitialDirectory = "c:\\";
   vybrat.Filter = "PowerPoint prezentace|*.ppt";
   vybrat.FilterIndex = 2;
   vybrat.RestoreDirectory = true;
 }
```

- Funkce *button1_Click()* je přiřazená tlačítku pro výběr prezentace. Po stisknutí se otevře komponenta průzkumník (OpenFileDialog) a v ní se označí naše vybraná prezentace. Po stisknutí OK se okno uzavře a dvakrát se načte absolutní cesta k vybrané prezentaci. Poprvé slouží jako parametr pro spuštění Power Pointu a podruhé pro zjištění jejího jména - to slouží pro hledání okna a poslání jména do PDA.

```
private void button1_Click(object sender, EventArgs e)
{
   int n = 0;
   DialogResult vysledek;
   vysledek = vybrat.ShowDialog();
   if (vysledek == DialogResult.OK)
   {
      pomocny = vybrat.FileName;
      filePPT = vybrat.FileName;
```

```
        n =  pomocny.LastIndexOf('\\');
        pomocny = pomocny.Substring(n + 1, pomocny.Length - n - 5);
        prezentace.Text = pomocny ;
    }

}
```

- Funkce *start_Click()* je přiřazená ke stisknutí tlačítka Start. Kontroluje se pomocí *GetPortNames()*, zda port s číslem, které zadal uživatel, skutečně existuje. Pokud existuje, pak se nastaví proměnná "potvrzeni" do log.1 a pokračuje se v provádění kódu funkce. Pokud port zadaného čísla není nalezen, vypíše se o tom report do stavového okna. Dále se vyplní parametry procesu - jméno aplikace, která je s procesem svázána (Power Point), argument spřažené aplikace, tedy pro nás jméno prezentace, které jsme získali výše. Pak se proces nastartuje = spustí se PowerPoint. Poté se inicializuje sériový port s parametry: číslo portu (zadáno uživatelem), bez parity, 8 datových bitů, 1 stop bit (toto nastavení je třeba mít i port v PDA aplikaci, aby komunikace správně fungovala). Potom sériový port otevřu. Dále následuje zablokování tlačítek, aby uživatel nemohl měnit parametry za běhu. Nakonec se do stavového okna vypíše hláška o zahájení odposlechu.

```
private void start_Click(object sender, EventArgs e)
{
    bool potvrzeni = false;
    string[] jmena = System.IO.Ports.SerialPort.GetPortNames();
    for (int x = 0; x < jmena.Length; x++)
    {
        if (jmena[x].Equals("COM"+portBlueTooth.Text))
        {
            potvrzeni = true;
        }
    }
    if (potvrzeni)
    {
        try
        {
            powerpoint.StartInfo.FileName = "POWERPNT.exe";
            powerpoint.StartInfo.Arguments = filePPT;
            powerpoint.Start();
            port = new SerialPort("COM" + portBlueTooth.Text, 19200, Parity.None, 8, StopBits.One);
            port.DataReceived += new SerialDataReceivedEventHandler(port_DataReceived);
            port.Open();
            vyber.Enabled = false;
            portBlueTooth.Enabled = false;
            ukoncitPoslech.Enabled = true;
            start.Enabled = false;
            stav.Text = "Poslech zahajen";
        }
        catch (Exception e2)
        {
            stav.Text="Chyba - problem s BT portem nebo aplikaci PowerPoint" + e2.Message;
        }
    }
    else
    {
        stav.Text = "Neplatne cislo portu!";
    }
}
```

- Zde je funkce *najitOkno()*, která vyhledá okno dle jména otevřené prezentace, a nastaví ho do

popředí, čímž se zajistí, že *SendKeys()* opravdu pošle zprávu správenému oknu. Tato funkce se volá před každým posláním, protože se aktivní okno může samozřejmě změnit. Problémem by mohlo být, pokud by se po nastavení okna s prezentací aktivním oknem jiné okno stalo aktivní ještě před posláním zprávy. Tato zpráva by se tedy doručila špatnému oknu a PowerPoint by nereagoval.

```
public void najitOkno()
{
    int handle = FindWindow(null, "Prezentace v aplikaci PowerPoint - ["+pomocny+"]");
    if (0 == handle)
    {
        stav.Text = "Nelze nalezt okno s prezentaci!";
    }
    else
    {
        SetForegroundWindow(handle);
    }
}
```

- funkce *port_DataRecieved()* je funkce, která se spustí, jakmile sériový port "port" přijme nová data. Pak se přečte jeden znak (více než jeden znak najednou není posílán) a porovnává se s podmínkami. Pokud je to písmeno 's', pak servisní deskotopová aplikace spustí prezentaci (pošle se simulované stisknutí tlačítka F5) a zpět do PDA pošle jméno prezentace. Pokud je přijmuté písmeno 'x', pak aplikace ukončí prezentaci. Ostatní posílaná písmena jsou pak přiřazena operacím v posuvu slidů v prezentaci:

'r' odpovídá posun v prezentaci o jeden slide dopředu (poslání stisku šipky vpravo)
'l' odpovídá posun v prezentaci o jeden slide dozadu (poslání stisku šipky vlevo)
'e' odpovídá přeskok na poslední slide (poslání stisku tlačítka End)
'f' odpovídá přeskok na první slide (poslání stisku tlačítka Home)

```
private void port_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    int prikaz = port.ReadChar();
    if (prikaz.Equals('s'))
    {
        SetForegroundWindow(FindWindow(null, powerpoint.MainWindowTitle));
        SendKeys.SendWait("{F5}");
        port.Write(pomocny);
        cas.ToLocalTime();
        stav.Text =  "Prezentace spustena";
    }
    else if (prikaz.Equals('x'))
    {
        najitOkno();
        SendKeys.SendWait("{ESC}");
        stav.Text = "Prezentace ukoncena";
    }
    else if (prikaz.Equals('r'))
    {
        najitOkno();
        SendKeys.SendWait("{RIGHT}");
    }
    else if (prikaz.Equals('l'))
    {
        najitOkno();
        SendKeys.SendWait("{LEFT}");
    }
    else if (prikaz.Equals('e'))
```

```
        {
            najitOkno();
            SendKeys.SendWait("{END}");
        }
        else if (prikaz.Equals('f'))
        {
            najitOkno();
            SendKeys.SendWait("{HOME}");
        }

    }
```

- *ukoncitPoslech_Click()* je funkce, která je přiřazena ke kliknutí na tlačítko pro ukončení poslechu. Pokud je port otevřen, tak se uzavře, ukončí se Power Point a odblokují se tlačítka pro změnu parametrů.

```
    private void ukoncitPoslech_Click(object sender, EventArgs e)
    {
        if (port.IsOpen)
        {
            port.Close();
        }
        if (!powerpoint.HasExited)
        {
            powerpoint.Kill();
        }
        vyber.Enabled = true;
        portBlueTooth.Enabled = true;
        start.Enabled = true;
        stav.Text = "Poslech ukoncen";
    }
  }
}
```

2.2.2 PDA část

Program, který posílá příkazy na desktop, je součástí aplikace obsahující i ovládání automatu na třídění kuliček. PDA část dálkového ovládání je opět velmi funkčně jednoduchá: pomocí onscreen tlačítek (či pomocí hardware buttons) uživatel nejdříve připojí PDA pomocí Bluetooth k PC a poté pustí prezentaci a jednoduše může listovat slidy. Název zobrazované prezentace se z PC pošle do PDA a vypíše se do okna. Po kliknutí na "ukončit prezentaci" se ukončí na PC Power Point.
PDA část je ještě jednodušší než desktopová - nejsložitější je zde tvorba GUI a nastavení portu.
Podívejme se teď opět detailně na zdrojový kód (pro značnou délku jsem ho zde zkrátil a vybral jen to podstatné) – soubor *„pwrPoint.cs“* z projektu PPTool:

- Z nestandardních Namespace používám speciální knihovnu Open .NET Compact Framework, která napravuje nedostatky standardního frameworku. Z této knihovny využívám Namespace IO.Serial - pro komunikace přes sériovou linku. Další důležité namespace jsou System.Runtime.InteropServices a Microsoft.WindowsCE.Forms pro ovládání prezentace pomocí hardware tlačítek.

```
using OpenNETCF.IO.Serial;
using Microsoft.WindowsCE.Forms;
using System.Runtime.InteropServices;
```

- Dva výčtové datové členy *KeyModifiers a KeysHardware* – modifikátory tlačítek a čísla, která

jsou přidělená uvnitř systému k jednotlivým HW tlačítkům. Pomocí těchto čísel identifikuji, které tlačítko bylo zmáčknuto.

- Třída *"myMessageWindow"* a *"RegisterHKeys"* slouží pro realizaci zaregistrování akce zmákčnutí hardwarového tlačítka k naší aplikaci. Dochází k importu funkcí ze základní knihovny "coredll.dll". Poté se registrují 4 HW tlačítka k naší aplikaci.

- Třída *"pwrPoint"* – hlavní třída realizující vlastní ovládání a komunikaci*:*
- Konstruktor třídy „pwrPoint“ (okna). Nejdříve se zavolá registrace hardwarových tlačítek na naši aplikaci – budeme, tak moci zachytávat události generované jejich stiskem. Dále se zavolá automaticky generovaný kód pro inicializaci komponent (až tak automaticky generovaný není, protože jsem ho manuálně editoval, abych mohl přiřadit handler při kliknutí na obrázek - ten pak funguje jako tlačítko). Dále se schová panel s nápovědou, zablokují se tlačítka, dokud se nepřipojí PDA k PC. Poté inicializujeme port: Bluetooth virtuální sériový port je v MyPal A716 označen jako "COM6". Dále nastavíme, že při každém přijatém bytu se vytvoří událost -> zavolá se handler DataReceived. Posílat se bude vždy po 1 bytu. A délka pole pro vstup je 999.

```
public pwrPoint()
        {
        this.messageWindow = new myMessageWindow(this);
        RegisterHKeys.RegisterRecordKey(this.messageWindow.Hwnd);
        InitializeComponent();
        panel1.Hide();
        zpet.Enabled = false;
        dopredu.Enabled = false;
        zacatek.Enabled = false;
        konec.Enabled = false;
        startStop.Enabled = false;
        zahaj.Enabled = false;
        portSettings = new HandshakeNone();
        port = new Port("COM6:", portSettings);
        port.RThreshold = 1;
        port.InputLen = 999;
        port.SThreshold = 1;
        port.DataReceived +=new Port.CommEvent(port_DataReceived);
        this.Closed+=new EventHandler(frmMain_Closed);
        }
```

- Funkce *ButtonPressed()* se zavolá při stisknutí tlačítka. Jako parametr se jí dá číslo tlačítka a podle něj se provede příslušná akce, tedy posun v prezentaci.

- Funkce pro připojení/odpojení PDA přes BT k PC - vyvolá se systémový dialog pro připojení. Zároveň se zablokují či odblokují příslušná tlačítka a změní barvy.

```
private void btnOpenClose_Click(object sender, System.EventArgs e)
        {
        if(port.IsOpen)
        {
         try
            {
            if(port.Close())
            {
             btnOpenClose.BackColor = Color.Green;
             btnOpenClose.Text = "Connect";
             startStop.Enabled = false;
             zahaj.Enabled = false;
             txtRx.Text = "Jmeno prezentace";
```

```
}
}
catch(Exception e2)
{
}
}
else
{
try
{
if(port.Open())
{
btnOpenClose.BackColor = Color.Red;
btnOpenClose.Text = "Disconnect";
startStop.Enabled = false;
zahaj.Enabled = true;
txtRx.Text = "";
}
}
catch(Exception e1)
{
}
}
}
```

- Handler pro událost přijetí dat. Načtou se byty na vstupu, překódují se do ASCII, vytvoří se odpovídající String a zobrazí se na obrazovce jako název prezentace.

```
private void port_DataReceived()
{
byte[] inputData = new byte[999];
inputData = port.Input;
Encoding enc = Encoding.ASCII;
displayString = enc.GetString(inputData, 0, inputData.Length);
txtRx.Text = displayString;
}
```

- Funkce přiřazená ke kliknutí na šipku doleva - poslání znaku 'l' = poslání znaku přes port, což znamená přiřadit pole bytů na "port.Output". Obdobné funkce jsou pro další tlačítka:

```
private void zpet_Click(object sender, System.EventArgs e)
{
byte[] outputData = new byte[1];
outputData[0] = Convert.ToByte('l');
port.Output = outputData;
}
```

- Funkce přiřazená ke kliknutí na šipku doprava - poslání znaku 'r'

```
private void dopredu_Click(object sender, System.EventArgs e)
{
byte[] outputData = new byte[1];
outputData[0] = Convert.ToByte('r');
port.Output = outputData;
}
```

- Funkce přiřazená ke kliknutí na tlačítko "OK" pro jít na poslední slide - poslání znaku 'e'

```
private void konec_Click(object sender, System.EventArgs e)
```

```
{
byte[] outputData = new byte[1];
outputData[0] = Convert.ToByte('e');
port.Output = outputData;
}
```

- Funkce přiřazená ke kliknutí na tlačítko "OK" pro jít na 1. slide - poslání znaku 'f'

```
private void zacatek_Click(object sender, System.EventArgs e)
{
byte[] outputData = new byte[1];
outputData[0] = Convert.ToByte('f');
port.Output = outputData;
}
```

- Funkce přiřazená ke kliknutí na tlačítko "Zavřít" - poslání znaku 'x' = ukončení prezentace a zablokování tlačítek pro posun v prezentaci.

```
private void button2_Click(object sender, System.EventArgs e)
{
byte[] outputData = new byte[1];
outputData[0] = Convert.ToByte('x');
port.Output = outputData;
zpet.Enabled = false;
dopredu.Enabled = false;
zacatek.Enabled = false;
konec.Enabled = false;
zahaj.Enabled = true;
startStop.Enabled = false;
txtRx.Text = "";
}
```

- Funkce přiřazená ke kliknutí na tlačítko "Zahájit" - poslání znaku 's' = zahájení prezentace a odblokují se tlačítka pro posuv v prezentaci.

```
private void zahaj_Click(object sender, System.EventArgs e)
{
byte[] outputData = new byte[1];
outputData[0] = Convert.ToByte('s');
port.Output = outputData;
zpet.Enabled = true;
dopredu.Enabled = true;
zacatek.Enabled = true;
konec.Enabled = true;
startStop.Enabled = true;
zahaj.Enabled = false;
}
```

- Funkce, která se zavolá při pokusu zavřít okno - pokud je port stále otevřen, nedovolí uzavřít okno a navrátit se do hlavního Menu aplikace.

```
private void frmMain_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
if(port.IsOpen)
{
 e.Cancel = true;
}
else
 PPTool.hlavniMenu.uvodni.Show();
}
```

Jak je vidět z popisu celého řešení, lze ho velmi jednoduše přestavět pro jakoukoli aplikaci, kterou lze ovládat příkazy z klávesnice - lze tedy lehce tímto SW ovládat WinAmp nebo prezentaci v Open Office atd.

2.3 Problémy a nevýhody řešení

2.3.1 Desktop

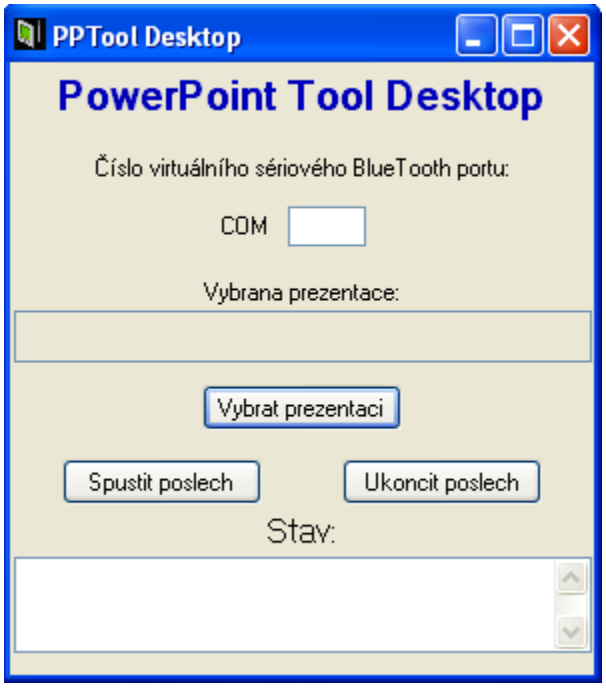
Podle mého názoru má mnou vytvořené řešení dálkového ovládání snad jen jeden významnější problém, a to je "Find window" - tedy hledání okna dle jména. Raději bych "handle" okna získal od připojené aplikace než touto kostrbatou cestou, bohužel na nic lepšího jsem nepřišel. Problémem je, pakliže by se okno s puštěnou prezentací v různých verzích Power Point jmenovalo jinak. Pak by bylo nutné aplikaci pro každou verzi upravit. Hledal jsem tedy lepší způsob, jak získat "handle", například pomocí přidružených procesů k našemu Power Point procesu, ale nic nevydalo kýžený výsledek, a tak jsem ponechal toto nedokonalé, ale v zásadě funkční řešení.
Problémem by také mohlo být, pokud by se po nastavení okna s prezentací do popředí jiné okno stalo aktivní ještě před posláním zprávy. Tato zpráva by se tedy doručila špatnému oknu a PowerPoint by nereagoval. Dalším problémem, na který jsem narazil a s kterým si moc neporadím, je problém názvů adresářů s mezerou - například: "Documents and settings". V takovém případě PowerPoint jako parametr nalezenou cestu k prezentaci nevezme a nenahraje ji.

2.4 Manuál uživatele

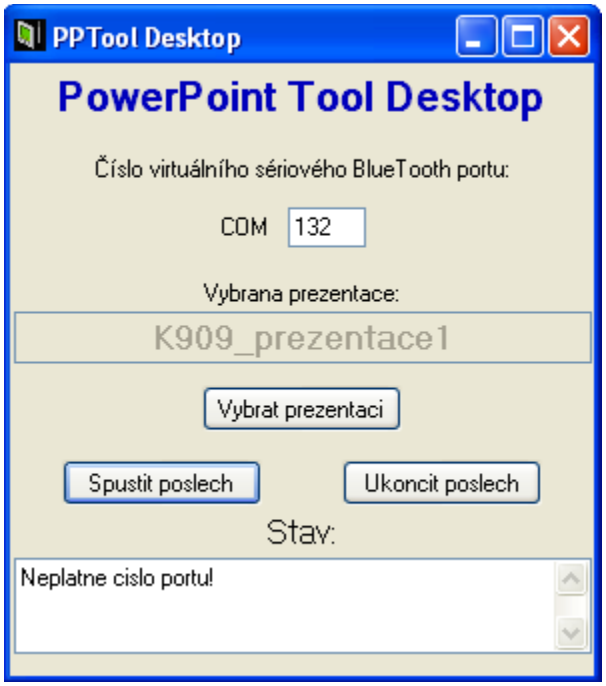
2.4.1 Desktop

K běhu je třeba mít nainstalovanou správnou verzi .Net Framework (lze stáhnout z webu Microsoft). Aplikace "PPTool Desktop" stačí zkopírovat kamkoli do paměti počítače. Po spuštění aplikace "PPTool Desktop.exe" je třeba zapsat číslo sériového COM portu, na který je namapován virtuální Bluetooth sériový port - viz nastavení Bluetooth adaptéru. Ten musí být vybaven SPP - "Serial Port Profile"- aby dokázal vytvořit virtuální port. Dále je třeba pomocí připojeného dialogu (průzkumníka) vybrat prezentaci pro spuštění ("Vybrat prezentaci"). Pozor! Zatím je problém s mezerami v absolutní cestě k prezentaci, proto je nejlepší vložit prezentaci do root (c:\) disku nebo do adresáře, ke kterému není cesta obsahující mezery. Po vybrání prezentace a vepsání čísla portu se tlačítkem "Zahájit poslech" spustí PowerPoint, otevře se sériový port a čeká se na připojení PDA, ze kterého přijdou příkazy pro zahájení prezentace a posuv v ní. Pro ukončení odposlechu slouží stejnojmenné tlačítko - zároveň se ukončí Power Point.

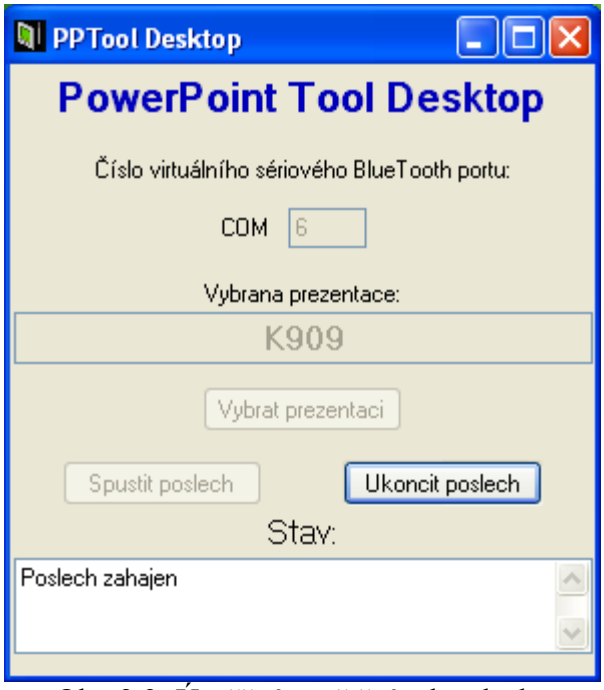
Screenshots:



Obr. 2.1: PPTool Desktop po spuštění



Obr. 2.2: Uživatel zadal neplatný port



Obr. 2.3: Úspěšné spuštění odposlechu

2.4.2 PDA

Před zahájením komunikace je třeba zapnout Bluetooth modul - zapnutí je indikované modrou stavovou diodou nad displejem. Bluetooth lze zapnout například v "Today screen" tlačítkem se znakem Bluetooth. Dále doporučuji předem spárovat PDA s PC, na kterém chceme ovládat prezentaci. Opět je nutno mít nainstalované správné verze knihoven OpenNET Smart Device Framework a .Net Compact Framework. Poté stačí nakopírovat .CAB s aplikací PPTool do paměti a po double-click se spustí instalace. Poté spustíme aplikaci PPTool. Před otevřením komunikace je také potřeba mít již spuštěnou servisní desktopovou aplikaci "PPTool Desktop.exe". Po spuštění aplikace "PPTool" se zobrazí hlavní obrazovka celé aplikace. Zde je na výběr ze tří tlačítek - pro dálkové ovládání je důležité tlačítko "Powerpoint". Po stisknutí tlačítka "Connect" se objeví obrazovka (dialog) s výběrem, k jakému zařízení PDA připojit - zobrazí se výsledky aktuálního průzkumu na Bluetooth frekvenci a spárovaná zařízení. Po úspěšném připojení k vybranému zařízení se uvolní tlačítko "Zahájit", které spustí prezentaci na PC. Zároveň se přijme a zobrazí název promítané prezentace. Spuštěním prezentace se odblokují tlačítka pro pohyb v prezentaci:

- "Jit na 1. slide" či "Jít na poslední slide" prezentace
- Šipka vpravo/vlevo - posun o 1 slide dopředu/dozadu
- "Ukončit" prezentaci - ukončí prezentaci

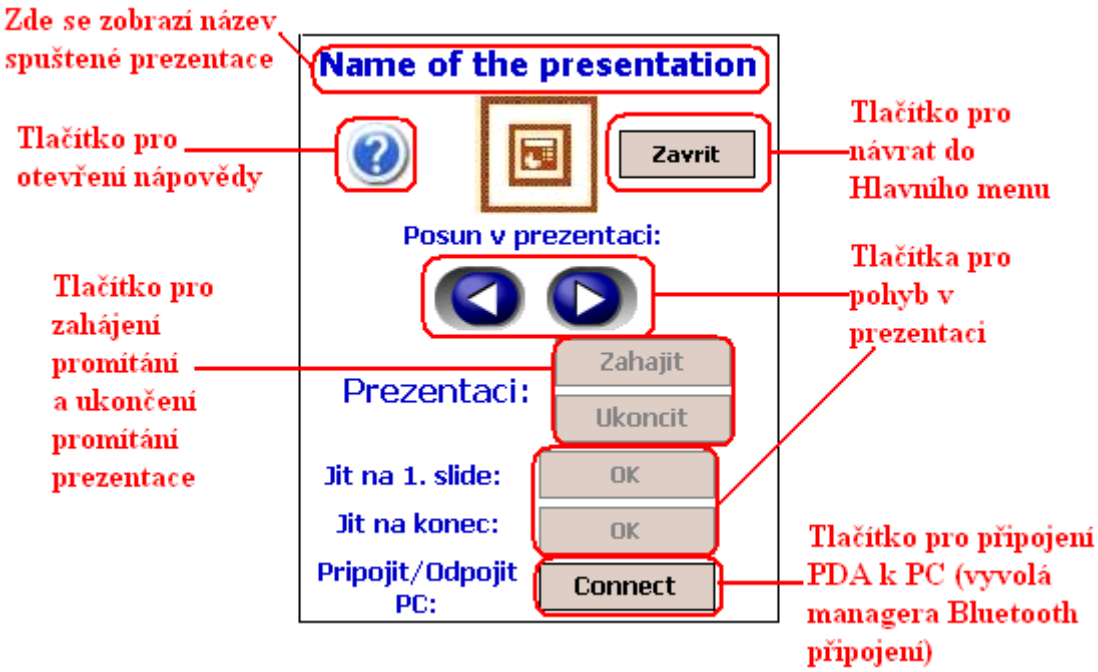
Zároveň lze ovládat prezentaci pomocí hardware tlačítek – viz obr 2.4.

Obr 2.4: Rozvržení ovládání na HW tlačítka

Po ukončení prezentace je třeba PDA odpojit od PC pomocí tlačítka "Disconnect".

Screenshots:

Obr. 2.5: Hlavní obrazovka aplikace pro PDA

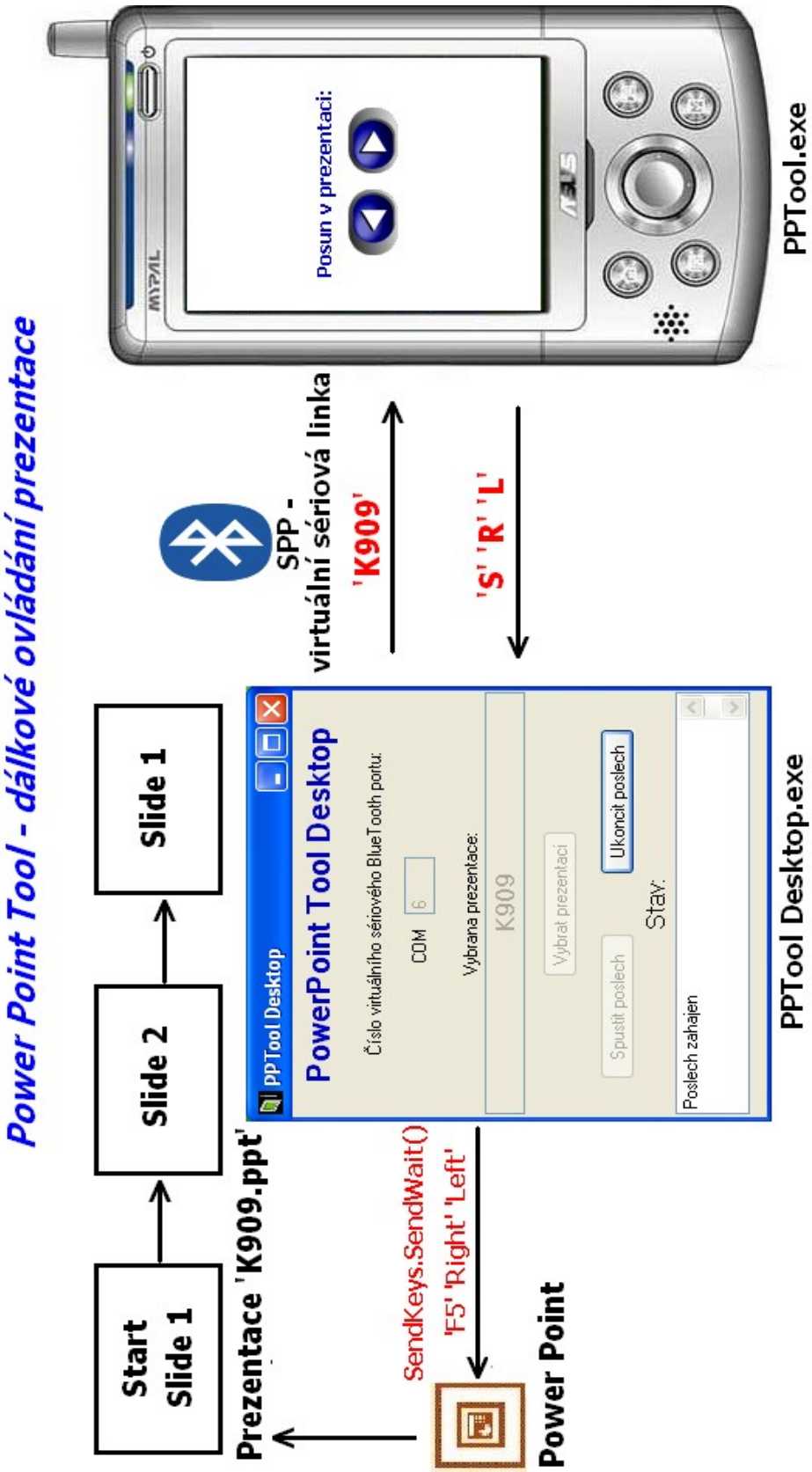


Obr. 2.6: Popis sekce pro vzdálené ovládání Power Point prezentace

2.5 Požadavky

- Bluetooth adaptér obsahující profil SPP (Serial Port Profile)
- Windows pro PC s nainstalovaným .Net Framework 2.0
- PDA se systémem PocketPC 2003 FE/SE s nainstalovaným .Net Compact Framework 1.0 a OpenNet Smart Device Framework 1.4 a vybavené BT modulem
- paměť 400kB

2.6 Schéma práce aplikace



3 Bezdrátový ovladač pro prezentaci modelu třídění kuliček

3.1 Úvod

Druhým a ve své podstatě hlavním bodem této bakalářské práce je návrh koncepce dálkového ovladače pro prezentaci činnosti některého modelu v laboratoři K09 s využitím PDA. Po prostudování funkčních modelů v laboratoři K09 jsem se rozhodl pro model "Pneumatické třídění kuliček" řízený PLC WAGO běžně používaný pro výuku ve vyšších ročnících. Tento model je zařazen do projektu Lablink[1]. Právě PLC WAGO je hlavním důvodem výběru tohoto modelu, protože lze tento programovatelný automat jednoduše dovybavit Bluetooth modulem, který stačí zasunout do konektoru pro servisní sériový kabel. Tím pádem nám opět Bluetooth port simuluje sériovou linku a celé řešení je obdobou řešení ovladače prezentace. Na straně PDA vybaveném Bluetooth čipem s SPP profilem je opět program, který otevře Bluetooth virtuální sériovou linku a posílá po ní jednoduchá data (zde číselné kódy), která jsou v PLC přijímána a interpretována jako příkazy pro otevření trysek, šoupat, fukaru či změnu režimu chodu (manuál/automat) etc. Nejsložitější byla tedy samotná realizace programu pro PLC WAGO.

3.2 Realizace

Realizace je stejně jako u ovladače prezentace rozdělena do dvou samostatných částí s tím rozdílem, že druhá část je program pro PLC WAGO.
Aplikace pro PDA je řešena v jazyce C# ve vývojovém prostředí Microsoft Visual Studio 2003 a program pro PLC WAGO v prostředí CoDeSys s využitím programování v žebříčkové logice a pomocí funkčních bloků.

3.2.1 PDA část

PDA část řešení je rozdělena na další dvě samostatné části - podprogram pro manuální řízení a podprogram pro spuštění a zastavení automatického třídění dle algoritmu nahraném do PLC.

Automatické třídění:
Toto je jednodušší část, na obrazovce jsou umístěny pouze tři tlačítka. První tlačítko připojí/odpojí PDA k/od WAGO PLC a zapne/restartuje automatické třídění, druhé vypne třídění a třetí vyvolá hlavní obrazovku. Podívejme se teď podrobněji na vybrané úseky zdrojového kódu – soubor *„autoKulicky.cs“* z projektu PPTool – třída s názvem *„autoKulicky“:*

- Z nestandardních Namespace používám opět již dříve popsaný OpenNET Smart Device Framework pro komunikaci po sériovém portu.

```
using OpenNETCF.IO.Serial;
```

- Konstruktor třídy - obrazovky (frame): kromě načtení obrázků a volání inicializace GUI se zde také nastavuje a inicializuje port - popis viz výše. Dále zde nastavuje počáteční logické hodnoty proměnných použitých pro řízení.

```
public class autoKulicky : System.Windows.Forms.Form
{
 public autoKulicky()
        {
                stopStav = false;
                stop.Enabled = false;
```

1 http://dce.felk.cvut.cz/lablink

```
            pripojitStav = false;
            portSettings = new HandshakeNone();
            port = new Port("COM6:", portSettings);
            port.RThreshold = 1;
            port.InputLen = 1;
            port.SThreshold = 1;
            port.DataReceived +=new Port.CommEvent(port_DataReceived);
        }
```

- *pripojit_Fce()* je funkce spojená s tlačítkem „Connect/Disconnect“ (Připojit/Odpojit). Pokud je port otevřen, tak se vyšle příkaz (kód 201), který resetuje proměnnou "autoRezim" - tedy uvede PLC do stavu, kdy není zapnut ani jeden režim. Počkám raději pomocí Thread.Sleep(), aby PLC mělo dost času zpracovat příkaz, pak vyšlu příkaz (kód 202) pro ukončení vykonávání algoritmu třídění - nahodím „STOP“ do log.1, dále se vyšle příkaz (kód 11) pro vypnutí tlakování kompresoru. Nakonec uzavřu port a nastavím vhodně barvy, obrázky tlačítek, texty a stavové proměnné. Pokud port není otevřen, tak se ho pokusím otevřít a vyšlu příkaz pro přechod PLC do programu automatického třídění a spustím tlakování kompresoru. Nakonec opět vhodně nastavím všechny stavové proměnné, barvy, ikony a texty.

```
        void pripojit_Fce()
        {
            if(port.IsOpen)
            {
                byte[] outputData = new byte[1];
                outputData[0] = 201;
                port.Output = outputData;
                System.Threading.Thread.Sleep(500);
                outputData[0] = 202;
                port.Output = outputData;
                stopStav = false;
                outputData[0] = 11;
                port.Output = outputData;

                try
                {
                    if(port.Close())
                    {
                        pripojit.Image = connect;
                        stop.Enabled = false;
                        stop.Image = zastavit;
                        pripojText.ForeColor = System.Drawing.Color.Lime;
                        pripojText.Text = "Pripojit a zapnout trideni:";
                        }
                }
            }
            else
            {
                try
                {
                    if(port.Open())
                    {

                        byte[] outputData = new byte[1];
                        outputData[0] = 203;
                        port.Output = outputData;
                        System.Threading.Thread.Sleep(500);
                        outputData[0] = 200;
                        port.Output = outputData;
                        pripojit.Image = disconnect;
```

```
                                        stop.Enabled = true;
                                        pripojText.ForeColor = System.Drawing.Color.Red;
                                        pripojText.Text = "Odpojit a vypnout trideni:";
                                        outputData[0] = 10;
                                        port.Output = outputData;
                                }
                        }
                }
        }
```

- *zastavit_Fce()* je funkce spřažená s druhým tlačítkem - Stop/Restart. Dle stavové proměnné "stopStav" zde vysílám příkazy k nastavení proměnné "Stop" v programu PLC, která zastaví či znovu spustí algoritmus třídění. Zároveň s tím měním vzhled GUI, aby měl uživatel přehled o situaci.

```
void zastavit_Fce()
{
        if(stopStav == false)
        {
                vypText.ForeColor = System.Drawing.Color.Lime;
                vypText.Text = "Restart trideni:";
                stop.Image = restartovat;
                byte[] outputData = new byte[1];
                outputData[0] = 202;
                port.Output = outputData;
                stopStav = true;
        }
        else
        {
                vypText.ForeColor = System.Drawing.Color.Red;
                vypText.Text = "Vypnout trideni:";
                stop.Image = zastavit;
                byte[] outputData = new byte[1];
                outputData[0] = 203;
                port.Output = outputData;
                stopStav = false;
        }
}
```

- *autoKulicky_Closing()* je funkce volaná při pokusu opustit „frame“ do hlavního Menu - pokud je port otevřen, tak je toto zakázáno - je to kvůli bezpečnosti.

```
private void autoKulicky_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
        if(port.IsOpen)
        {
         e.Cancel = true;
        }else
                PPTool.hlavniMenu.uvodni.Show();
}
```

Manuální třídění:

Manuální řízení spočívá v tom, že uživatel připojí PDA v WAGO PLC modelu "Třídění kuliček" a dál pomocí kliknutí na příslušné tlačítko na obrazovce ovládá jednotlivé akční členy modelu jako trysky, šoupě, fukar a kompresor. Správně by PDA mělo přijímat údaj o barvě právě zpracovávaného míčku, ale tuto funkci se mi nepodařilo úspěšně rozběhnout. Podívejme se teď na vybrané úseky zdrojového kódu – soubor *„trideniKulicek.cs“* z projektu PPTool – třída s názvem „*trideniKulicek*“:

- Použité Namespace a nastavení portu jsou shodné s Automatickým tříděním. V konstruktoru se mimo nastavení portu nastaví i logické stavové proměnné do počátečního stavu false.
- *pripojit_Click()* je funkce, která je volána při zmáčknutí tlačítka pro připojení/odpojení k/od PLC WAGO (tlačítko s logem Bluetooth). Nejdříve se otestuje, zda je port již otevřen. Pokud ano, nastaví se stavové proměnné všech akčních členů na log 1 a zavolají se jejich obslužné funkce. Tyto obslužné funkce způsobí vyslání signálů pro deaktivaci příslušných akčních členů (vypnutí trysky, kompresoru atd.). Před každým voláním raději počkám vždy půl sekundy, aby PLC mělo čas reagovat. Dále se pokusím uzavřít port a pozmění se GUI, aby odpovídalo novému stavu a zablokují se tlačítka ovládající akční členy.

```
private void pripojit_Click(object sender, System.EventArgs e)
        {
                if(port.IsOpen)
                {
                        davkovaciTryskaStav = true;
                        michaciTryskaStav = true;
                        pneuTryska1Stav = true;
                        pneuTryska2Stav = true;
                        pneuTryska3Stav = true;
                        pneuTryska4Stav = true;
                        fukarStav = true;
                        soupeStav = true;
                        compresorStav = true;
                        soupe_Fce();
                        System.Threading.Thread.Sleep(500);
                        fukar_fce();
                        System.Threading.Thread.Sleep(500);
                        pneuTryska3_Fce();
                        System.Threading.Thread.Sleep(500);
                        pneuTryska4_Fce();
                        System.Threading.Thread.Sleep(500);
                        pneuTryska2_Fce();
                        System.Threading.Thread.Sleep(500);
                        pneuTryska1_Fce();
                        System.Threading.Thread.Sleep(500);
                        davkovaciTryska_Fce();
                        System.Threading.Thread.Sleep(500);
                        michaciTryska_Fce();
                        System.Threading.Thread.Sleep(500);
                        compresor_Fce();
                        System.Threading.Thread.Sleep(500);
                        byte[] outputData = new byte[1];
                        outputData[0] = 101;
                        port.Output = outputData;
                        try
                        {
                                if(port.Close())
                                {
                                        pripojit.Image = pripojitObr;
                                        zavrit.Enabled = true;
                                        davkovaciTryska.Enabled = false;
                                        michaciTryska.Enabled = false;
                                        pneuTryska1.Enabled = false;
                                        pneuTryska2.Enabled = false;
                                        pneuTryska4.Enabled = false;
                                        pneuTryska3.Enabled = false;
                                        fukar.Enabled = false;
                                        soupe.Enabled = false;
                                        compresor.Enabled = false;
```

```
                }
            }
        }
```

- Pokud PDA ještě není připojeno, pak se vyvolá systémový dialog pro připojení a po úspěšném připojení k PLC se odblokují ovládací tlačítka a nastaví se opět všechny logické stavové proměnné do log 1. Následně se zavolají obslužné funkce. Toto se provádí proto, že model může být po předchozím používání v různém stavu (akční členy mohou být sepnuté) a já požaduji, aby byl model v klidu. Proto si jejich deaktivaci takto pojistím. Jediný akční člen, který je po připojení programem zapnut, je kompresor - tedy jeho tlakování.

```
        else
        {
            try
            {
                if(port.Open())
                {
                    pripojit.Image = odpojitObr;
                    byte[] outputData = new byte[1];
                    outputData[0] = 100;
                    port.Output = outputData;
                    davkovaciTryska.Enabled = true;
                    zavrit.Enabled = false;
                    michaciTryska.Enabled = true;
                    pneuTryska1.Enabled = true;
                    pneuTryska2.Enabled = true;
                    pneuTryska4.Enabled = true;
                    pneuTryska3.Enabled = true;
                    fukar.Enabled = true;
                    soupe.Enabled = true;
                    compresor.Enabled = true;
                    davkovaciTryskaStav = true;
                    michaciTryskaStav = true;
                    pneuTryska1Stav = true;
                    pneuTryska2Stav = true;
                    pneuTryska3Stav = true;
                    pneuTryska4Stav = true;
                    fukarStav = true;
                    soupeStav = true;
                    compresorStav = false;
                    soupe_Fce();
                    System.Threading.Thread.Sleep(500);
                    fukar_fce();
                    System.Threading.Thread.Sleep(500);
                    pneuTryska3_Fce();
                    System.Threading.Thread.Sleep(500);
                    pneuTryska4_Fce();
                    System.Threading.Thread.Sleep(500);
                    pneuTryska2_Fce();
                    System.Threading.Thread.Sleep(500);
                    pneuTryska1_Fce();
                    System.Threading.Thread.Sleep(500);
                    davkovaciTryska_Fce();
                    System.Threading.Thread.Sleep(500);
                    michaciTryska_Fce();
                    System.Threading.Thread.Sleep(500);
                    compresor_Fce();

                }
            }
```

- Funkce *port_DataReceived()* se volá při příchodu dat do PDA z PLC. Funkce by měla dle číselné hodnoty přijatého bytu nastavit ve vizualizaci správnou barvu právě zpracovávaného míčku. Bohužel zpětná komunikace nefunguje jak by měla.

```
private void port_DataReceived()
{
        byte[] inputData = new byte[1];
        inputData = port.Input;
        byte cer = 3;
        byte bil = 0;
        byte cen = 1;
        byte mod = 2;
        byte yel = 4;
        if(inputData[0].Equals(bil))
        {
         barva.Image = white;
        }
        else if(inputData[0].Equals(cen))
        {
         barva.Image = black;
         }
         else if(inputData[0].Equals(cer))
        {
         barva.Image = red;
        }
        else if(inputData[0].Equals(mod))
        {
         barva.Image = blue;
        }
        else if(inputData[0].Equals(yel))
        {
        barva.Image = yellow;
        }
}
```

- Funkce *davkovaciTryska_Fce()* je volána při kliknutí na tlačítko příslušející ovládání dávkovací trysky modelu (dávkování míčků k senzorům). Přesně takováto funkce, jen se změněnými číselnými kódy operací, které se vysílají, a obrázky, které zobrazují sepnutí/rozepnutí, je pro každý akční člen modelu (trysky, šoupě, fukar, kompresor) - je jich tedy devět. Dle příslušné stavové proměnné se vybere správná operace (číselný kód) a ten se vyšle přes virtuální sériovou linku. Dále se zneguje stavová proměnná a změní se obrázek tlačítka ve vizualizaci ON/OFF.

```
void davkovaciTryska_Fce()
{
        if(davkovaciTryskaStav == false)
        {
                byte[] outputData = new byte[1];
                outputData[0] = 20;
                port.Output = outputData;
                davkovaciTryskaStav=true;
                davkovaciTryska.Image = tlacitkoOff;}
        else
        {
                byte[] outputData = new byte[1];
                outputData[0] = 21;
                port.Output = outputData;
                davkovaciTryskaStav = false;
                davkovaciTryska.Image = tlacitkoOn;}
```

```
}

private void davkovaciTryska_Click(object sender, System.EventArgs e)
{
        davkovaciTryska_Fce();
}
```

- Při pokusu vrátit se do hlavní obrazovky se opět nejdříve zkontroluje, zda není port otevřen. Pakliže ano, pak se nepovolí opustit obrazovku, dokud uživatel neukončí komunikaci.

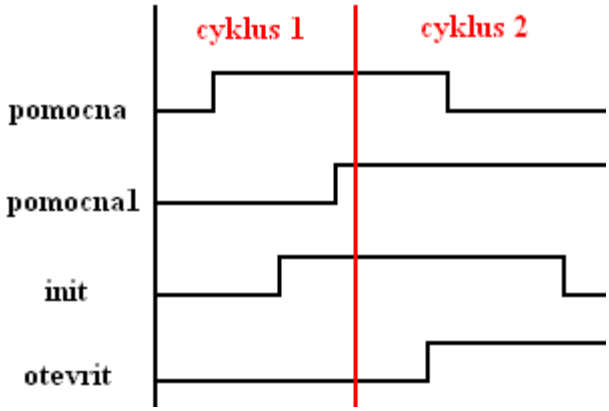
```
private void trideniKulicek_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
        if(port.IsOpen)
        {
                e.Cancel = true;
        }
        else
        PPTool.hlavniMenu.uvodni.Show();
}
```

3.2.2 WAGO část

Pro vývoj aplikace pro PLC WAGO je využito vývojového prostředí CoDeSys ve verzi 2.3.4.7. Program je rozdělen do čtyř částí (programů), z nichž dvě jsou vytvořeny žebříčkovou logikou a dvě metodou funkčních bloků. Základem programu je soubor „*manual.pro*“ z webu[2] projektu Lablink a pro automatické třídění byl Ing. Pavlem Burgetem dodán program obsahující funkční algoritmus. Z knihoven využívám: *Util.lib*, *Standard.lib* a především *SerComm.lib* a *Serial_Interface.lib,* které zajišťují celou komunikaci. Zdrojový kód je přiložen: "*program_pro_wago.pro*"

PLC_PRG:
Toto je hlavní program automatu, který běží v nekonečné smyčce (princip programovatelného automatu). Program je psán žebříčkovou logikou. Prvních pět řádků (příček) slouží k určení barvy aktuálně zpracovávaného míčku pomocí výstupů senzorů. Další řádek (č.6) volá podprogram BT, který zabezpečuje celou komunikaci s PDA přes Bluetooth. Výstupy z podprogramu BT se řídí další tři řádky (č. 7,8,9), kde se aktivuje buď program obhospodařující manuální řízení automatu nebo program s naprogramovanou logikou automatického třídění. Tento algoritmus se spustí nastavením logické proměnné „Start“ při spuštění automatického režimu. Řádky 10 až 14 slouží k automatické inicializaci virtuálního sériového portu představovaného blokem SERIAL_INTERFACE v podprogramu BT. Inicializace využívá dvou cyklů PLC (z důvodu časově správné změny log. proměnných): V prvním cyklu se nastaví logická proměnná "pomocna" na TRUE (protože "pomocna1" má hodnotu FALSE). Stav proměnné "pomocna" je kopírován do proměnné "init" - v prvním cyklu tedy TRUE. Nakonec je nastavena na TRUE i proměnná "pomocna1". V druhém cyklu je proměnnou "pomocna1" nastavena proměnná "pomocna" na FALSE a "otevrit" na TRUE. Tím pádem je i "init" FALSE. Tím je dokončena inicializace sériového portu a je připraven k posílání a příjmu dat. Grafické znázornění na obr. 3.10.

2 http://dce.felk.cvut.cz/lablink/tutorials/colorsorter/manualne.pro



Obr. 3.10: Průběh proměnných při inicializaci sériového portu

BT:
BT je program zabezpečující komunikaci s PDA přes Bluetooth. BT je naprogramován pomocí jazyka funkčních bloků. Na řádku číslo 1 je umístěna komponenta SERIAL_INTERFACE s názvem "komunikace", představující celý sériový port. Nastavení atributů bloku:

- bCOM_PORT_NR je číslo portu, který otevírám. Pro nás je to číslo 1 = servisní port, ve kterém je umístěn Bluetooth adaptér.
- cbBAUDRATE je počet baudů/sekundu = 19200 bd/s. Toto nastavení společně s cpPARITY, csSTOPBITS, cfFLOW_CONTROL a cbsBYTESIZE musí odpovídat atributům přenosu nastaveným v PDA.
- cpPARITY je nastavení parity - PARITY_NO tedy bez parity
- csSTOPBITS = počet stopbitů (zde 1 stopbit).
- cbsBYTESIZE = počet datových bitů (zde 8)
- cfFLOW_CONTROL = řízení toku dat (zde zakázáno)
- iBYTES_TO_SEND = počet bytů k poslání z WAGO do PDA (zde 1 byte). S tímto směrem komunikace se ale vyskytl problém. Viz kapitola 3.3.
- ptSEND_BUFFER = adresa prvního bytu bufferu obsahující byty k poslání z WAGO PLC.
- xSTART_SEND = proměnná ovládající posílání dat z WAGO PLC
- utRECEIVE_BUFFER = buffer pro přijatá data
- xINIT a xOPEN_COM_PORT slouží k inicializaci a otevření portu. Přísluší jim globální proměnné "init" a "otevrit" (viz výše).

Řádek číslo dva a tři představují přepínač mezi režimy manuálního a automatického chodu. Přečte se poslední přijatý byte ze sériového portu (odkaz na pole BUFFER dle INDEX, který ukazuje na další buňku pole, do které se bude zapisovat) a porovná se s přiděleným číselným kódem (100 = zapnout manuální režim,101 = vypnout manuální režim,200 = zapnout automatický režim,201 = vypnout automatický režim) pomocí bloku EQ. Pokud je výsledek TRUE, pak se nastaví pomocí klopného SR obvodu do proměnných "manualRezim"a "autoRezim" správné hodnoty. Dále je zde obdobný řádek (č.4) pro spuštění tlakování kompresoru (to musí být puštěno co nejdříve, protože trvá několik minut kompresor natlakovat, a proto posílá tento příkaz mezi prvními) a řádek č.5 pro vypnutí třídícího algoritmu obsaženého v automatickém režimu. Řádek 6-10 slouží k uložení barvy aktuálně zpracovaného míčku do pole - pokud je logická proměnná představující barvu pravdivá, pak se uloží do pole její přidělený číselný kód, jinak v poli zůstane původní hodnota. Bohužel s posíláním barev do PDA je problém viz kapitola 3.3.

MANUAL:
Podprogram, který zajišťuje manuální řízení modelu - tedy převádí příkazy ze sériové linky na otevírání ventilů, šoupat atd. MANUAL, je taktéž napsán pomocí funkčních bloků. Samotný

program se skládá z osmi principiálně stejných řádků: opět je zde porovnávání posledního přijatého bytu s kódem a SET nebo RESET pomocí klopného SR obvodu dané výstupní proměnné. Je to stejné jako nastavování režimu v podprogramu BT. Přiřazení číselných kódů k operacím, tedy otevření a uzavření trysky, šoupěte či zapnutí/vypnutí fukaru, je zcela jasné z implementace programu - viz samotný přiložený kód.

AUTO:
Podprogram AUTO, který je napsán žebříčkovou logikou, obsahuje implementaci algoritmu automatického třídění kuliček dle jejich barvy - toto je jedním z úkolů v předmětech vyšších ročníků - viz pokyny na webu Lablink[3]. Já jsem obdržel od p.ing. Burgeta již hotovou implementaci. Stačilo jí tedy zakomponovat do celkového řešení. Pro řízení automatického třídění slouží globální logické proměnné "Start" (pokud je v log1 - tak se třídění odstartuje a běží) a "Stop" (pokud je v log 1., pak se vykonávání programu automatického třídění zastaví).

3.3 Problémy a nevýhody řešení

Největší, pro mne nepřekonatelný, problém (strávil jsem s ním mnoho hodin práce) je zpětné posílání dat z PLC do PDA přes BT linku.
Použití tohoto zpětného přenosu je v posílání barvy míčku, který byl právě zpracován senzory - v podstatě tedy jen drobnost, ale z principu mne nenalezení řešení mrzí.
Problémem je automatizace posílání 1 bytu dat - ruční inicializace přenosu z PLC do PDA v prostředí CoDeSys je bez problému, byte PDA přijme a zobrazí dle konstrukce Switch správnou barvu. Ruční poslání spočívá v manuální změně stavu binární proměnné 'pomo' na logickou 1. PLC pak vyšle byte. Problémem je tedy tuto operaci zautomatizovat. O to jsem se pokoušel různými konstrukcemi. Všechny bohužel více či méně neúspěšné (pošlou se data náhodně, nebo se způsobí zamrznutí celého programu PLC). Achillovou patou je potřeba mít na vstupu xSTART_SEND bloku SERIAL_INTERFACE (zajišťuje celou komunikaci) proměnnou, kterou může sám blok měnit.
Jde o to, že když se proměnná na vstupu změní na log 1 z log 0, inicializuje se poslání x bytů (x = vstup iBytes_to_send), kde adresa počátečního bytu je na vstupu ptSEND_BUFFER bloku
a blok SERIAL_INTERFACE změní stav této proměnné zpět na log. 0. Jak jsem již psal, i přes mnohahodinové úsilí se mi rozumné řešení zpětné komunikace nepodařilo nalézt.

Druhým problémem, který ovšem nemám možnost řešit, je automatizace připojení PDA k PLC - tedy aby se uživatel nemusel neustále proklikávat systémovým managerem Bluetooth připojení a vybírat port atd.

3.4 Manuál uživatele

Nahrání a spuštění programu na PLC:

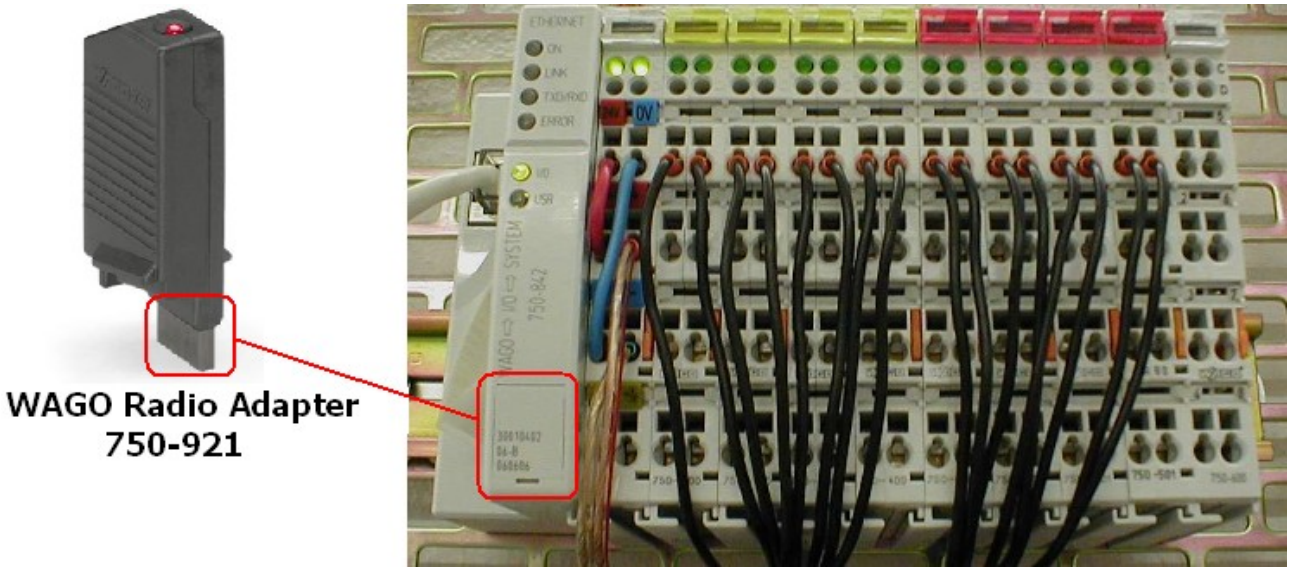
Pro nahrání programu do PLC budeme potřebovat nainstalované vývojové prostředí CoDeSys (testováno s verzí 2.3.4.7). V něm otevřeme soubor s kódem: "*program_pro_wago.pro*". Dále zkontrolujeme v menu "Online" v záložce "Communication Parameters" parametry pro připojení k cílovému PLC WAGO 750-842. Správné parametry by měly být již přednastaveny, ale přesto pro jistotu je zde uvedu:
IP: 147.32.87.217
Port number: 2455
Transport protocol: tcp
Debug level: 16#0000
Pokud je s parametry vše v pořádku,klikneme v menu "Online" na "Login" a vyčkáme na připojení. Zde je třeba podotknout, že nejdříve musíme mít platnou rezervaci na určitý čas na "Color sorter"

3 http://dce.felk.cvut.cz/lablink/tutorials/colorsorter/navod.htm

v systému Lablink a být zalogováni na "Color sorter" - pak počítač (jeho IP), ze kterého jsme se zalogovali přes Lablink, je zplnomocněný se přes CoDeSys se k PLC na jeho IP připojit. Pokud jsme toto vše učinili, tak bychom měli být bez problému zalogováni z CoDeSys na PLC. Pokud jsme tedy ve stavu Online, pak klikneme na "Download" a nahrajeme aktuální program (tedy náš). Pak už stačí jen kliknout na "Run" , program je spuštěn a můžeme připojit PDA.

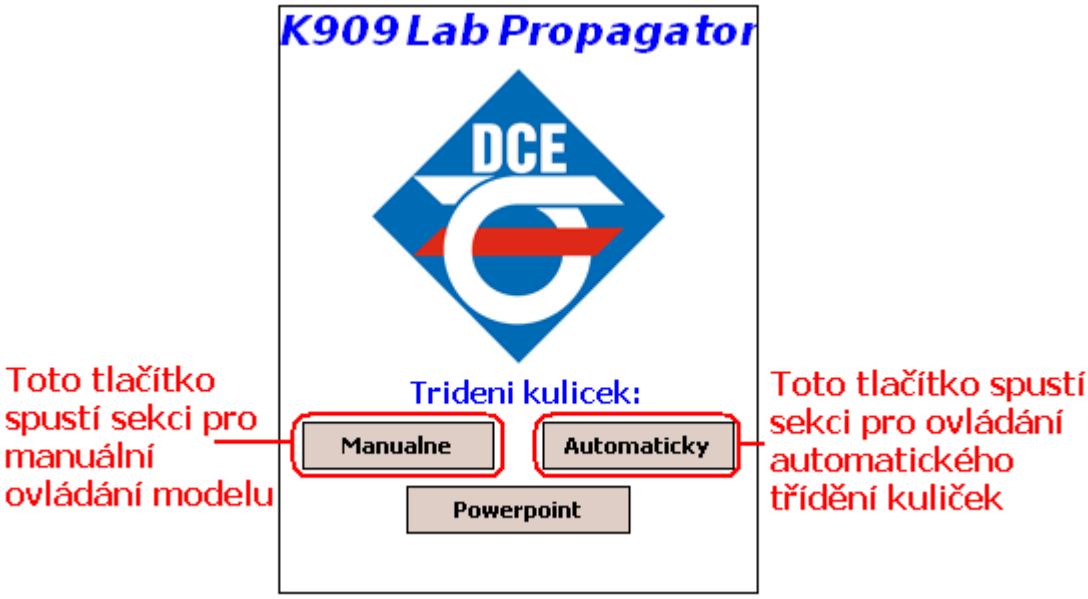
Připojení PDA:

Nejdříve musíme zapojit Bluetooth adaptér (WAGO Radio Adapter 750-921) do servisního portu viz obrázek 3.11. Celá jeho instalace spočívá v pouhém zastrčení do slotu. Poté je již vše připraveno na připojení PDA přes Bluetooth.



Obr 3.11: Propojení adaptéru a PLC

Na PDA nainstalujeme (viz kapitola 2.4.2) a spustíme program PPTool.exe. V hlavní obrazovce máme na výběr z tlačítek "Manualne" (spustí se obrazovka s manuálním ovládáním modelu) a "Automaticky", které vyvolá obrazovku s ovládacími prvky pro automatické třídění.

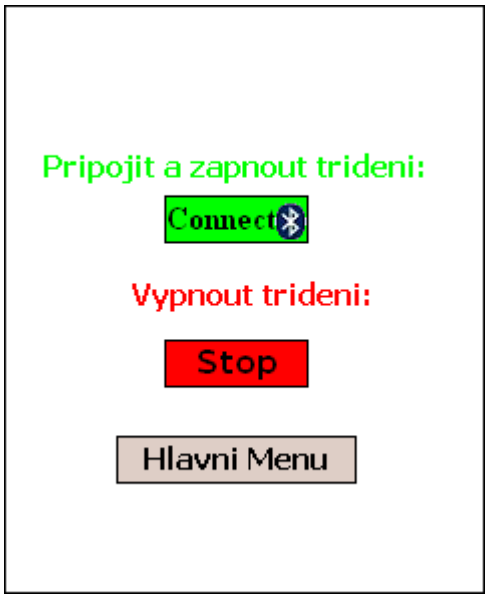


Obr 3.12: Hlavní obrazovka

Automatické třídění:

Pokud vybereme "Automaticky", tak se nám otevře obrazovka se třemi tlačítky:

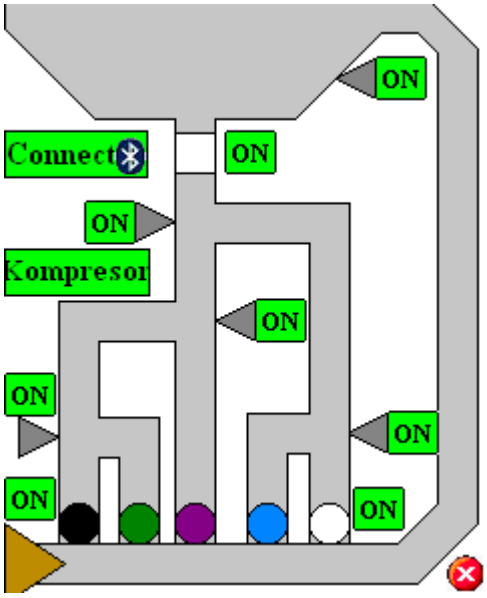
- "Connect/Disconnect" - připojí PDA k PLC WAGO a spustí algoritmus automatického třídění. Pří druhém kliknutí pak ukončí třídění a odpojí PDA.
- "Stop/Restart" - vypne třídění, či ho opět spustí.
- "Hlavni Menu" - vrátí uživatele do hlavní obrazovky, ale to jen v případě, když je PDA odpojeno.



Obr 3.13: Automatické třídění

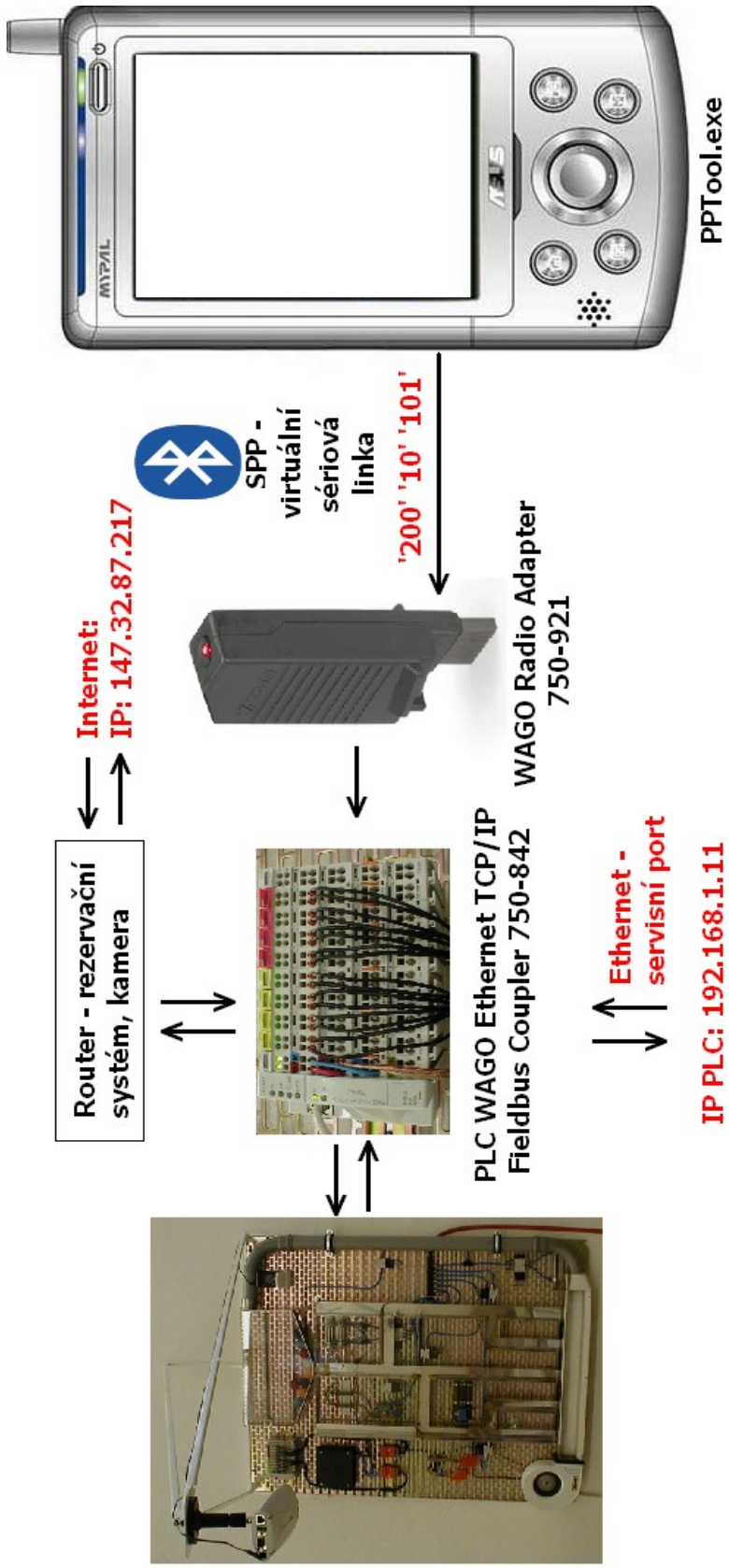
Manuální ovládání:

Pokud vybereme "Manualne", pak se otevře obrazovka s vyobrazením modelu a jeho akčních členů v podobě tlačítek ON/OFF. Nejdříve je třeba pomocí tlačítka "Connect" připojit pomocí již známého Bluetooth dialogu. Po úspěšném připojení se odblokují tlačítka ovládající akční členy. Po ukončení práce je pak před opuštěním obrazovky tlačítkem s křížkem nutno pomocí tlačítka "Disconnect" odpojit PDA, jinak je pokus o opuštění zamítnut.



Obr 3.14: Manuální ovládání

3.5 Schéma práce řešení



3.6 Požadavky

- PDA stejně vybavené jako v bodě 2.5
- model "Pneumatické třídění kuliček" řízený PLC WAGO Ethernet TCP/IP Fieldbus Coupler 750-842
- Bluetooth adaptér pro WAGO. Označení: WAGO Radio Adapter 750-921

4 Shrnutí

Oba hlavní úkoly, tedy návrh a realizace dálkového ovladače modelu v laboratoři K09 a dálkového ovladače prezentace v Power Point v podobě PDA, se mi podařilo splnit. Oba ovladače fungují uspokojivě, i když je nutno přiznat, že obě řešení nejsou zcela ideální a dalo by se najít ještě mnoho vylepšení (viz kapitoly 2.3 a 3.3). Mimo jiné by bylo přínosné přeprogramovat celou PDA část v novém Visual Studio 2005 s novým .NET Compact Framework 2.0, který plně nahrazuje nutnost využívání OpenNet Smart Device Framework. Především by pak stálo za to rozšířit počet modelů v K09 možných ovládat přes nějakou bezdrátovou technologii (ať už BlueTooth či WiFi) – zvláště zajímavé by pak bylo řídit pomocí PDA model vzducholodi taktéž umístěný v laboratoři K09.

5 Literatura a software

- Projekt Lablink *http://dce.felk.cvut.cz/lablink* Katedra řízení FEL ČVUT - dokumentace k modelu třídění barevných míčků a zdrojových kód využitý jako základ pro program pro PLC WAGO *manual.pro*
- *OpenNETCF.org* využití knihoven a ukázkového zdrojového kódu "OpenNETCF.IO.Serial Samples" jako základ pro sériovou komunikaci: *http://www.opennetcf.org/samples/SerialCSharp.zip*
- Web *www.ce4you.cz* - specifikace PDA Asus MyPal A716
- Využití ukázkového kódu pro sériovou komunikaci "SerialPort Terminal" od Noah Coad http://coad.net

6 Přiložené soubory

Na přiložené CD jsou tyto přílohy:

- Program pro PLC WAGO *„program_pro_wago.pro“*
- Program pro *„PDA PPTool_PPC.ARMV4.cab“* (instalační soubor .CAB)
- Potřebné knihovny pro PDA (instalační soubory .CAB):

  „netcf.all.wce4.ARMV4.cab“

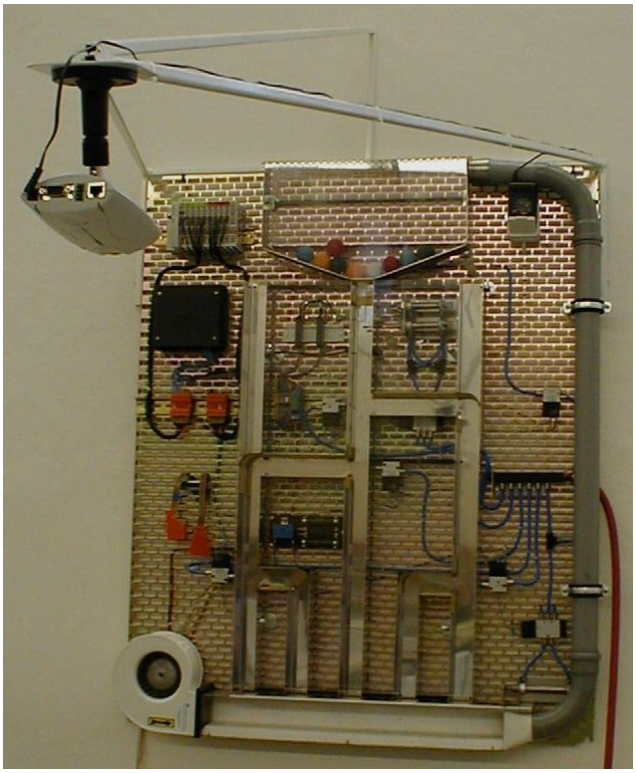
  „OpenNETCF.SDF.ppc3.armv4.cab“
- Program pro PC *„PPTool Desktop.exe“*
- Text bakalářské práce ve formátu PDF, SWX
- Dále jsou přiloženy celé projekty ve Visual Studiu 2003 a 2005

Přílohy

A Popis modelu

Převzato z webu projektu Lablink[4]:

Model třídícího mechanismu (Obr 3.1) byl zkonstruován za účelem demonstrace algoritmu třídění ping-pongových míčků podle jejich barvy. Vzhledem k možnému využití modelu pro účely distančního vzdělávání byl model navrhnut tak, aby v zapnutém stavu nevyžadoval žádnou vnější manuální obsluhu. Do počátečního stavu se celý mechanismus dostane pouhým vyfoukáním míčků do horního zásobníku. Tím je splněna základní podmínka pro vzdálené řízení bez nutného zásahu člověka přímo na místě.



Obr. 3.1: Model třídícího mechanismu

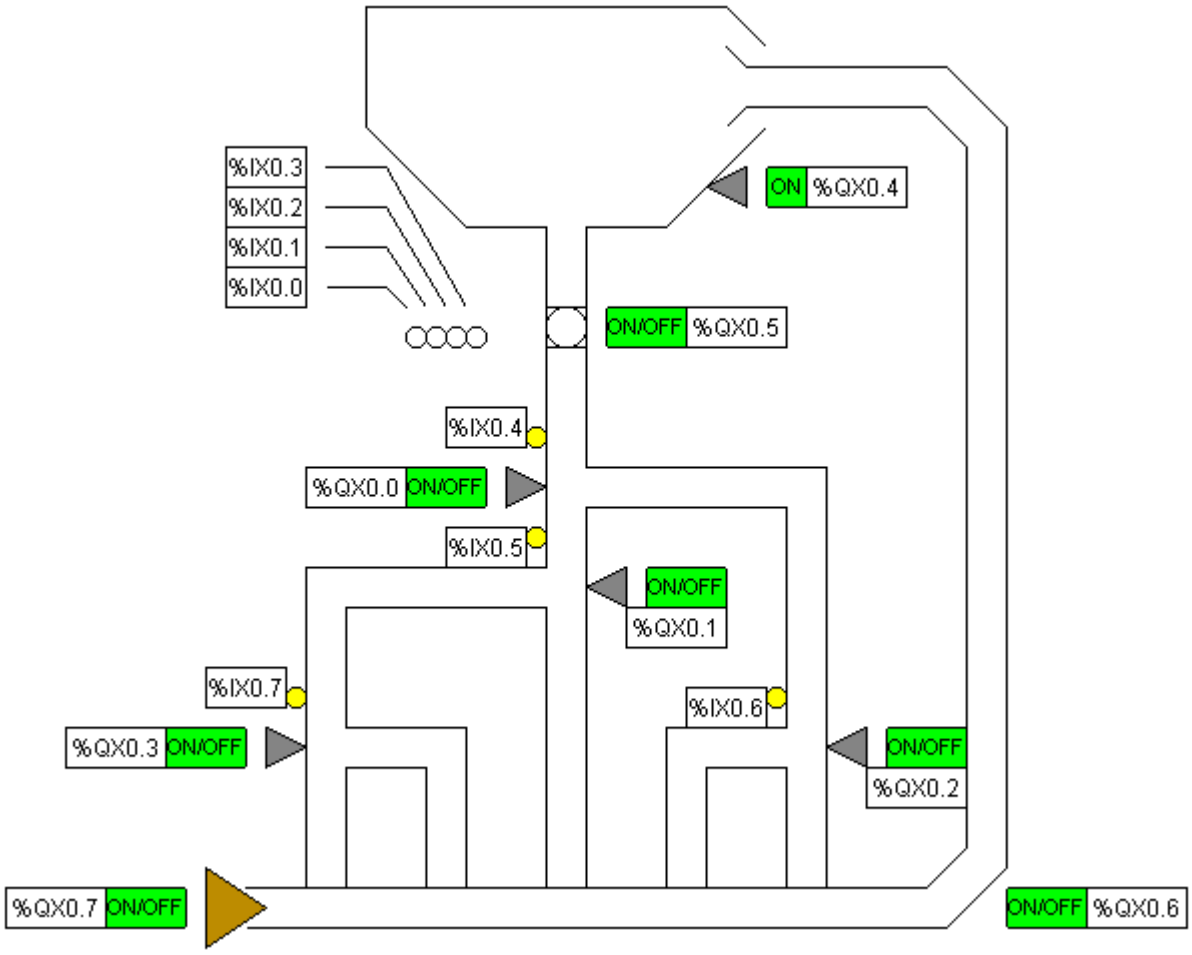
Popis systému

Míčky jsou tříděny do pěti šachet podle jejich barvy (černá, modrá, žlutá, červená, černá). Míčků je celkem deset, každá barva je tedy zastoupena dvěma míčky. Proces třídění spočívá ve dvou základních procedurách:

- identifikace barvy míčku
- vychylování míčku při volném pádu v reakci na průchod optickým čidlem umístěným před vychylovací tryskou.

Systém má osm digitálních vstupů a stejný počet digitálních výstupů. Logická 0 odpovídá 0 V, logická 1 odpovídá 24 V. Polohu a adresy jednotlivých snímačů a akčních členů zapojených do PLC WAGO ukazuje Obr 3.2.

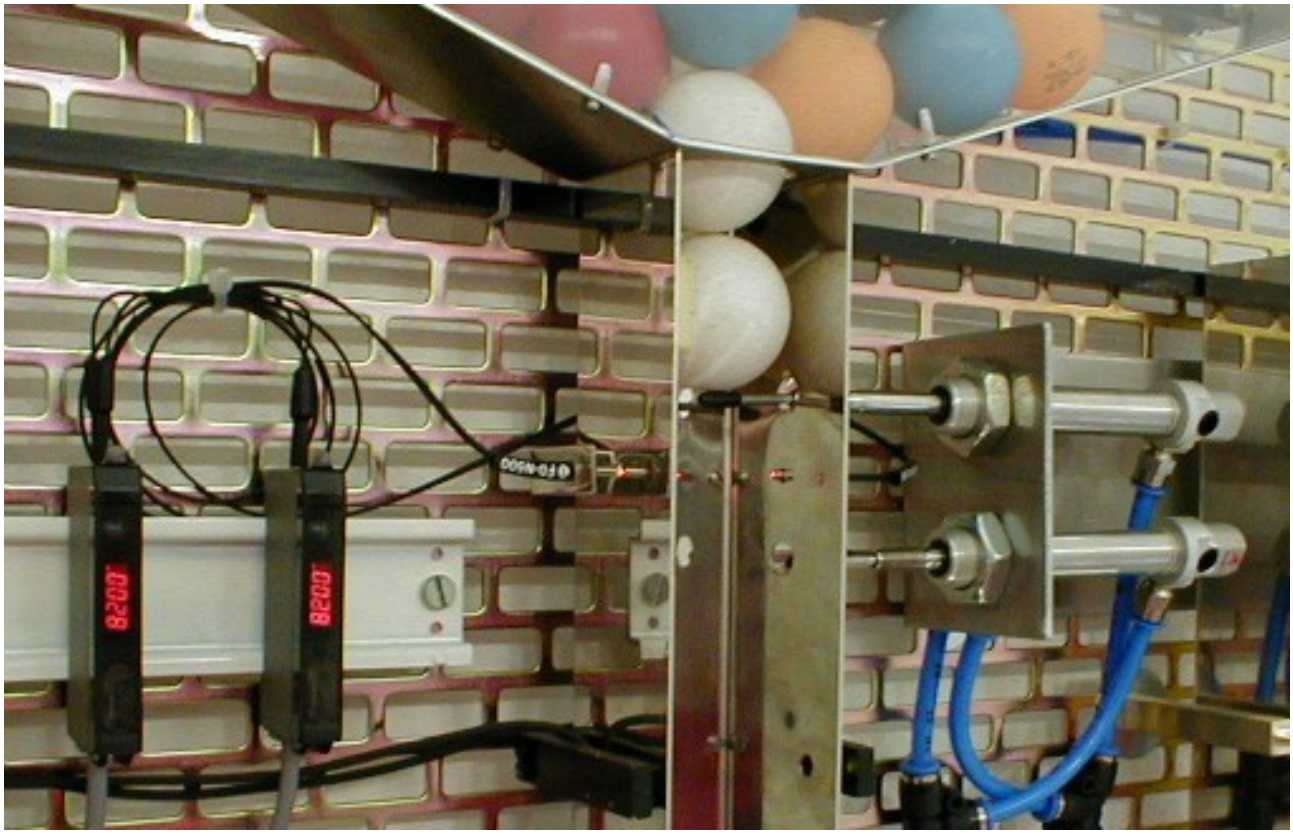
4 http://dce.felk.cvut.cz/lablink/tutorials/colorsorter/popis.htm



Obr 3.2: Poloha a adresy snímačů a akčních členů

Vstupy

Barva míčku je rozpoznávána pomocí dvou nezávislých optických čidel FX-D1P (Obr 3.3). Tato čidla permanentně vysílají modulovaný světelný paprsek červené barvy, který se odráží od pozorovaného předmětu. Každá barva odráží světelný paprsek s jinou intenzitou. Obecně platí, že nejméně odráživé jsou černé míčky, naopak nejvíce paprsku odrážejí bílé míčky. Odražený signál je přiveden optickým kabelem zpět do zesilovače a ten podle úrovně odraženého signálu nastaví své výstupy. Každý zesilovač má dvě nastavitelné prahové úrovně. Je-li intenzita odraženého paprsku menší než obě prahové úrovně, jsou oba výstupy OUT1 i OUT2 zesilovače rovny logické 0. Překročí-li odražený paprsek první (menší) prahovou úroveň, zesilovač nastaví do logické 1 výstup OUT1 a překročí-li paprsek i druhou (větší) prahovou úroveň, nastaví do logické 1 i svůj druhý výstup OUT2. U jednotlivých zesilovačů lze nastavit tři různé modulační frekvence vysílaného světelného paprsku, takže se čidla navzájem neovlivňují a nejsou závislá na okolním osvětlení.



Obr 3.3: Snímače rozpoznávající barvu

Každý zesilovač má dva výstupy (OUT1 a OUT2), celkem je tedy možné rozpoznat pět barev. Následující tabulka ukazuje hodnoty výstupů z optických čidel v jednotlivých případech barvy. Výstupy jsou zapojeny do vstupů %IX0.0 až %IX0.3 PLC WAGO.

Hodnoty výstupů snímačů barvy:

Barva	%IX0.0	%IX0.1	%IX0.2	%IX0.3
Černá	0	0	0	0
Zelená	1	0	0	0
Fialová	1	1	0	0
Bledě modrá	1	1	0	1
Bílá	1	1	1	1

Vstupy %IX0.4 až %IX0.7 jsou jednotlivé infračervené závory, které indikují průchod míčku závorou (Obr 3.4). Příslušný vstup je v době průchodu míčku roven logické 1, jinak je roven logické 0.

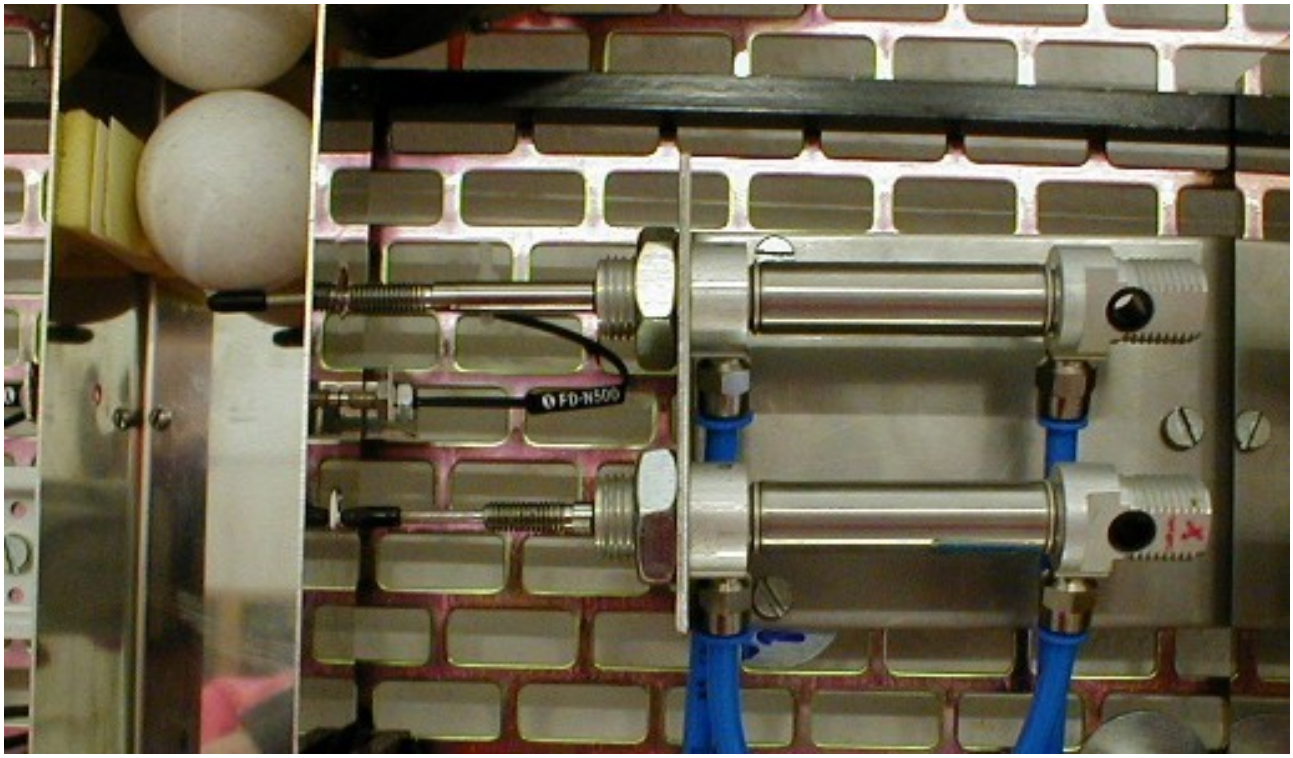
Obr 3.4: Infračervené závory

Výstupy

Výstupy %QX0.0 až %QX0.3 jsou jednotlivé pneumatické trysky firmy FESTO (Obr 3.5), které se spouští přivedením logické 1 (24 V). Je-li na trysce nastavena logická 0 (0 V), tryska nefouká. Výstup %QX0.4 je pneumatická tryska stejného typu jako předchozí čtyři vychylovací trysky, slouží však k promíchávání ještě neroztříděných míčků v horním zásobníku. Aktivita každé trysky je indikována LED diodou zelené barvy umístěnou poblíž trysky.

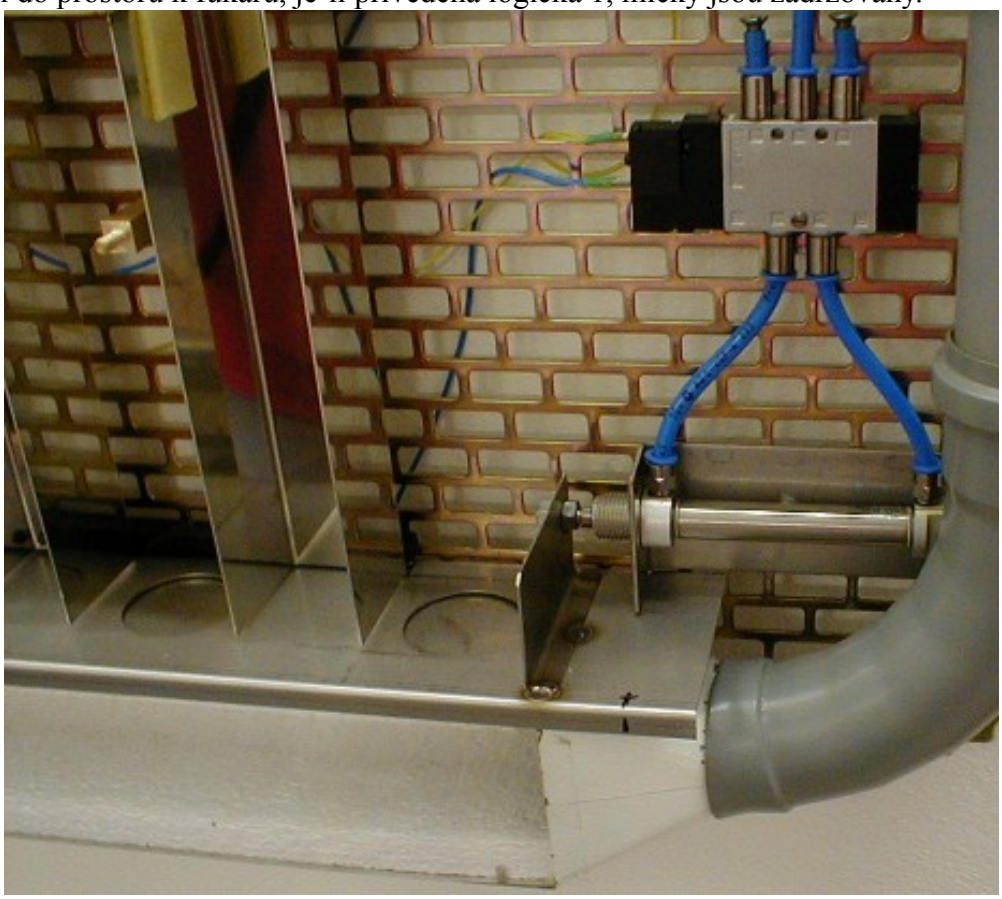
Obr 3.5: Pneumatická tryska FESTO

Výstup %QX0.5 je dvoucestný pneumatický ventil FESTO (Obr 3.6), který slouží k zadržování míčků při rozpoznávání barvy. Je-li na něj přivedena logická 1, je míček zadržen, v opačném případě padá volným pádem.



Obr 3.6: Dvoucestný pneumatický ventil FESTO

Výstup %QX0.6 je opět pneumatický dvoucestný ventil (Obr 3.7), který slouží k manipulaci s šoupětem, které zavírá a otvírá jednotlivé šachty. Je-li na něj přivedena logická 0, míčky propadávají do prostoru k fukaru, je-li přivedena logická 1, míčky jsou zadržovány.

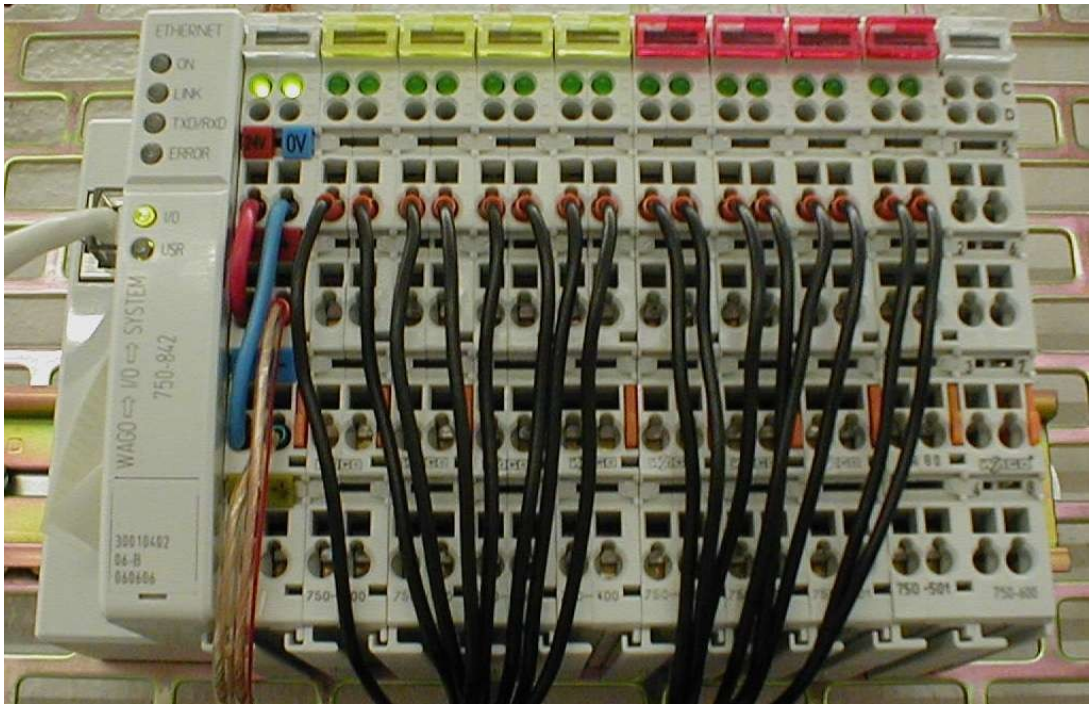


Obr 3.7: Spodní šoupě

Výstup %QX0.7 je elektrický fukar (Obr 3.8) sloužící k vyfoukávání míčků zpět do horního zásobníku. Fukar pracuje na napětí 230 V, proto je spínán pomocí relé. Při vyfoukávání je nutné mít zavřené šoupě (%QX0.6 = 1), protože proud vzduchu z fukaru by unikal do šachet a nebyl by dostatečně silný k vyfouknutí všech míčků do horního zásobníku. Aktivitu fukaru indikuje LED dioda modré barvy umístěná poblíž fukaru.

Obr 3.8: Fukar

Navíc je k PLC připojeno ovládání kompresoru na výstupu %QX0.8. Nastavením tohoto výstupu do log. 1, například zápisem compressor AT %QX0.8:BOOL:=TRUE; začne tlakování nádrže. Jakmile tlak dosáhne určité hodnoty, která je nutná pro správný provoz ventilů a trysek FESTO, je tato skutečnost signalizována na vstupu %IX0.8 (log. 1) a zapne se osvětlení modelu. Natlakování kompresoru trvá asi 10 minut. Pokud by tlak klesl pod minimální požadovanou úroveň, bude model od tlaku odpojen, osvětlení zhasne a na vstupu %IX0.8 se objeví log. 0.
Důležitá poznámka:
Výstup %QX0.9 v logické 1 způsobí reset PLC neboli kompletní smazání programu.
Systém je řízen programovatelným logickým automatem WAGO Ethernet TCP/IP Programmable Fieldbus Controller 750-842 (Obr 3.9) a sledován webkamerou AXIS 2100.



Obr 3.9: PLC WAGO Ethernet TCP/IP Fieldbus Coupler 750-842