PRAXE

č. 7	STŘEDNÍ PRŮMYSLOVÁ ŠKOLA CHOMUTOV	David Herko
8. 10. 2024	VÝTAH	V4

Obsah

Zadání

S využitím IO karty sestavte v jazyku C# program pro ovládání modelu výtahu. Dle svého uvážení zvolte buď prostředí konzolové aplikace nebo grafické prostředí.

Výsledné řešení musí obsahovat následující funkcionalitu:

- Spolehlivá inicializace výchozího stavu zařízení

- Ovládání výtahu v rozsahu min. čtyř pater z kabinových a patrových tlačítek

- Zpracování informací jednotlivých čidel modelu (patro, podlahový spínač apod.) a reakce podobná s reálným zařízením spolu s indikací stavu výtahu pomocí patrové indikace

- Implementace alespoň základní bezpečnosti provozu během pohybu kabiny

Teorie

Výtah je napájen 12V / 1A, což se nastavuje na zdroji ke kterému je výtah připojen. Programuje se programuje přes IO kartu a v jazyce C#. K propojení se využívá knihovna `Automation.Bdaq4`. Výtah je aktivní v logické 0. Výtah umí jezdit nahoru a dolů. Na každém patře je přivolávací tlačítko a LEDky, které se mají rozsvěcet podle toho, kam výtah jede. Ve výtahu je žárovka a podlahový spínač pro kontrolu, jestli ve výtahu někdo je. Na výtahu je sedmi-segmentový displej, který by měl ukazovat číslo patra, na kterém se výtah nachází.

Popis řešení

Výstupem programu je konzolové okno, ve kterém jsou vypisovány různé (většinou debugové) akce. Program používá mojí pomocnou C# knihovnu `Dejvoss.BitShift` pro operace s bitovými posuny. Po spuštění programu se void `Main()` napojí na zařízení a načte XML soubor s konfigurací. Vytvoří se instance class `Controller`, `Elevator`, `ElevatorController` a `Buttons`. Každá ovládá nějaké zařízení či jeho část, viz sekce rozbor metod níže. Pak už následuje hlavní loop, který se opakuje do ukončení programu. V něm se ze začátku získá několik proměnných potřebných k chodu výtaku - zmáčknutá přivolávací tlačítka (jak na patrech, tak v kabině) ve formě čísla, boolean jestli je výtah otevřený, boolean jestli se právě stav výtahových dveří změnil, bool jestli někdo ve výtahu stojí a bool jestli právě někdo do výtahu přišel nebo z něj odešel. Pokud výtah stojí a není na žádném patře, spustí se void `recall()` který se pokusí výtah dopravit do nejbližšího patra. Pokud se stav dveří změnil, světlo se buď rozsvítí nebo zhasne, podle toho jestli se otevřely nebo zavřely. Pokud do výtahu někdo nastoupil, světlo zůstane svítit. Pokud ne, zhasne se. A ve finále program zkontroluje, jestli bylo zmáčknuto nějaké ovládací tlačítko. Podle toho výtah odjede do cílového patra. A zpátky na začátek hlavního loopu.

Rozbor proměnných a metod

Program.cs – hlavní část programu

int currentFloor	ukládá současnou pozici výtahu
bool isGoing	ukládá, jestli je výtah v pohybu
void Main()	hlavní vstupní část kódu

Controller.cs – základní ovládací prvky IO karty

InstantDoCtrl ioDO, InstantDiCtrl ioDI	ovladače io karty
byte[] portData	pro jednodužší čtení z portů se všechny změny ukládají sem
int port1out, port2out, port3in, port4in	čísla input/output portů
void write(int port, byte input)	uloží byte hodnotu [input] do portu [port]
void read(int port)	přečte byte hodnotu z portu [port]

Elevator.cs – ovládá jednotlivé funkce výtahu

void reset()	nastaví výchozí hodnoty io karty
void engineOn() / void engineOff()	zapne / vypne motor výtahu
void engineUp() / void engineDown()	nastaví směr výtahu nahoru / dolu
void lightOn() / void lightOff()	zapne / vypne světlo v kabině
void ledUpOn() / void ledUpOff()	zapne / vypne směrové LEDky na patrech (směr nahoru)
void ledDownOn() / void ledDownOff()	zapne / vypne směrové LEDky na patrech (směr dolů)
void playSound(int ms)	pustí zvuk pípnutí na [ms] milisekund
int whereIs()	vrátí číslo patra, kde se výtah nachází
bool isDoorOpen()	vrátí boolean, jestli jsou dveře otevřené
bool isFloorSensorOn()	vrátí boolean, jestli je čidlo ve výtahu aktivní (někdo tam je)

ElevatorController.cs – spojuje funkce z Elevator.cs do větších celků

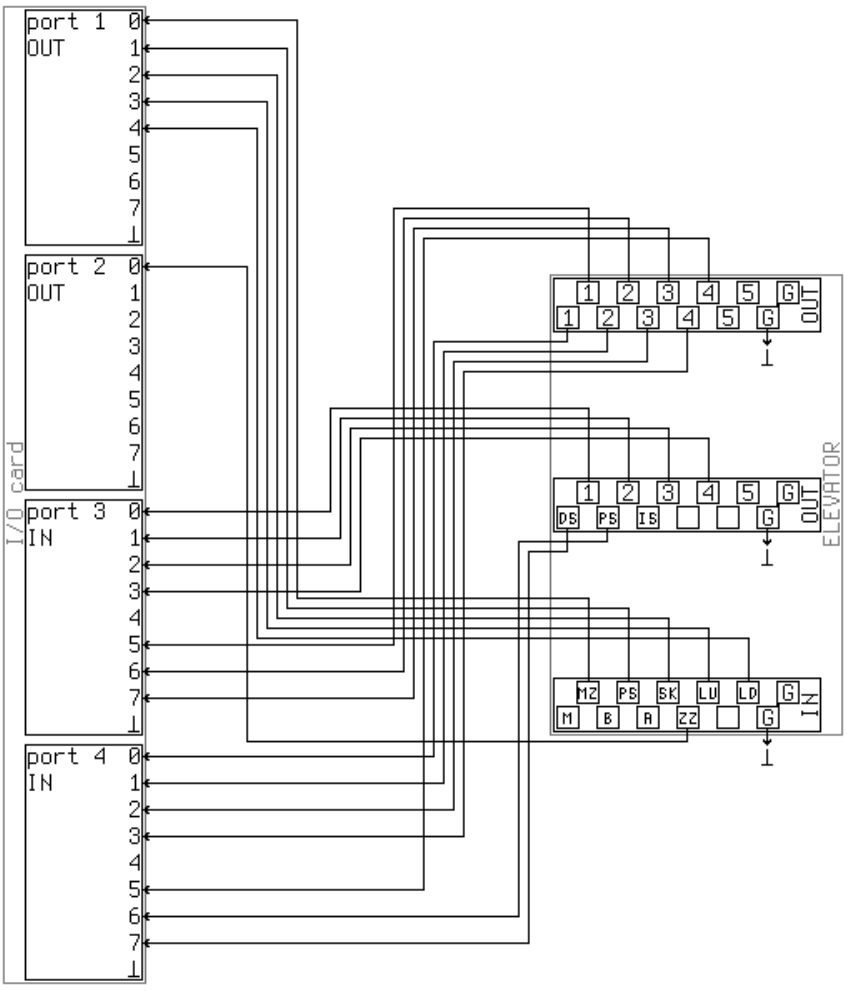
bool isGoing	je výtah v pohybu?
bool prevDoorStatus	ukládá předchozí stav dveří
bool prevFloorStatus	ukládá předchozí stav podlahového čida

void goUp() / void goDown()	rozjede výtah směrem nahoru / dolu
void stop()	zastaví výah
int getCurrentFloor()	získá pozici výtahu
bool isOnFloor()	vrátí true, když je výtah na patře
void goToFloor(int targetFloor)	výtah pojede na patro číslo [targetFloor]
void recall()	výtah se pokusí jet na nejbližší patro
bool doorChanged()	kontroluje, jestli se stav dveří změnil
bool floorSensorChanged()	kontroluje, jestli se stav podlahového čidla změnil

Buttons.cs

int floorButtonPressed()	vrátí číslo patra, na kterém bylo zmáčknuto tlačítko
int cabinButtonPressed()	vrátí číslo patra, na které chce člověk ve výtahu jet
int getFloorButtons()	spojuje předchozí funkce pro jednodušší použití

Schéma zapojení



Komentovaný výpis kódu

Program.cs

```
using Automation.BDaq;
using System;

namespace Herko.Elevator
{
    internal class Program
    {
        public int currentFloor = 0;
        public bool isGoing = false;

        static void Main(string[] args)
        {
            //console title
            Console.Title = "Elevator controller v0.3-dev | (c) DEJVOSS Productions 2024";

            //device info
            DeviceInformation di = new DeviceInformation();
            di.Description = "PCIE-1730,BID#0";
            di.DeviceMode = AccessMode.ModeWrite;

            InstantDoCtrl ioDO = new InstantDoCtrl();
            ioDO.SelectedDevice = di;
            ioDO.LoadProfile("profile.xml");

            InstantDiCtrl ioDI = new InstantDiCtrl();
            ioDI.SelectedDevice = di;
            ioDI.LoadProfile("profile.xml");

            //create needed instances
            Controller c = new Controller(ioDO, ioDI);
            Elevator elevator = new Elevator(c);
            ElevatorController ec = new ElevatorController(elevator);
            Buttons buttons = new Buttons(c);

            Console.WriteLine("[MAIN] initialized!");

            //main loop
            while (true)
            {
                //getting variables every tick to work with
                int floorButtonPressed = buttons.getFloorButtons();

                bool doorChanged = ec.doorChanged();
                bool isDoorOpen = elevator.isDoorOpen();

                bool floorChanged = ec.floorSensorChanged();
                bool isOccupied = elevator.isFloorSensorOn();

                //recalls to floor if not one one
                if (!ec.isGoing && ec.getCurrentFloor() == -1)
                {
                    ec.recall();
                }

                //if door status changed
                if (doorChanged)
                {
                    //turn lights on or off
                    if (isDoorOpen)
                        elevator.lightOn();
                    else if (!isDoorOpen && !isOccupied)
```

```
                    elevator.lightOff();
                }

                //if floor sensor changed
                if (floorChanged)
                {
                    //turn lights on or off
                    if (isOccupied)
                        elevator.lightOn();
                    else if (!isDoorOpen && !isOccupied)
                        elevator.lightOff();
                }

                //checks for button presses
                if (floorButtonPressed != -1 && !ec.isGoing && !isDoorOpen)
                {
                    Console.WriteLine("called elevator to floor " + floorButtonPressed);
                    ec.goToFloor(floorButtonPressed);
                }
            }

            Console.ReadLine(); //for testing DO NOT REMOVE
        }
    }
}
```

Controller.cs

```
using Automation.BDaq;

namespace Herko.Elevator
{
    internal class Controller
    {
        //controllers
        public InstantDoCtrl ioDO;
        public InstantDiCtrl ioDI;

        //bytes to work with
        public byte[] portData = new byte[4];

        //ports
        public int port1out = 0;
        public int port2out = 1;
        public int port3in = 0;
        public int port4in = 1;

        //construct
        public Controller(InstantDoCtrl ioDO, InstantDiCtrl ioDI)
        {
            this.ioDO = ioDO;
            this.ioDI = ioDI;
        }

        //write [input] to [port]
        public void write(int port, byte input)
        {
            ioDO.Write(port, input);
            portData[port] = input;
        }

        //reads byte at [port]
        public byte read(int port)
        {
```

```
            byte data;
            ioDI.Read(port, out data);

            return data;
        }
    }
}
```

Elevator.cs

```
using Dejvoss.BitShift;
using System;
using System.Threading;

namespace Herko.Elevator
{
    internal class Elevator
    {
        //board controller
        public Controller c;

        //ACTIVE LOG.0!!!!!
        public Elevator(Controller c)
        {
            this.c = c;
        }

        //set defaults so everything is off on start
        public void reset()
        {
            c.write(c.port1out, 0b11111111);
            c.write(c.port2out, 0b11111111);
        }

        //turn on engine
        public void engineOn()
        {
            c.write(c.port1out, EightBit.setAt(c.portData[c.port1out], 0, 0));
            Console.WriteLine("[ELEVATOR] engine on");
        }

        //turn off engine
        public void engineOff()
        {
            c.write(c.port1out, EightBit.setAt(c.portData[c.port1out], 0, 1));
            Console.WriteLine("[ELEVATOR] engine off");
        }

        //set elevator direction to up
        public void engineUp()
        {
            c.write(c.port1out, EightBit.setAt(c.portData[c.port1out], 1, 1));
            Console.WriteLine("[ELEVATOR] going up");
        }

        //set elevator direction to down
        public void engineDown()
        {
            c.write(c.port1out, EightBit.setAt(c.portData[c.port1out], 1, 0));
            Console.WriteLine("[ELEVATOR] going down");
        }

        //turn cabin light on
        public void lightOn()
```

```
        {
            c.write(c.port1out, EightBit.setAt(c.portData[c.port1out], 2, 0));
            Console.WriteLine("[ELEVATOR] cabin light on");
        }

        //turn cabin light off
        public void lightOff()
        {
            c.write(c.port1out, EightBit.setAt(c.portData[c.port1out], 2, 1));
            Console.WriteLine("[ELEVATOR] cabin light off");
        }

        //turn direction indicator UP lights on
        public void ledUpOn()
        {
            c.write(c.port1out, EightBit.setAt(c.portData[c.port1out], 3, 0));
            Console.WriteLine("[ELEVATOR] led up on");
        }

        //turn direction indicator UP lights off
        public void ledUpOff()
        {
            c.write(c.port1out, EightBit.setAt(c.portData[c.port1out], 3, 1));
            Console.WriteLine("[ELEVATOR] led up off");
        }

        //turn direction indicator DOWN lights on
        public void ledDownOn()
        {
            c.write(c.port1out, EightBit.setAt(c.portData[c.port1out], 4, 0));
            Console.WriteLine("[ELEVATOR] led down on");
        }

        //turn direction indicator DOWN lights off
        public void ledDownOff()
        {
            c.write(c.port1out, EightBit.setAt(c.portData[c.port1out], 4, 1));
            Console.WriteLine("[ELEVATOR] led down off");
        }

        //beeps for [ms] ms
        public void playSound(int ms)
        {
            c.write(c.port2out, EightBit.setAt(c.portData[c.port2out], 0, 0));
            Thread.Sleep(ms);
            c.write(c.port2out, EightBit.setAt(c.portData[c.port2out], 0, 1));

            Console.WriteLine("[ELEVATOR] played sound");
        }

        //returns floor number of elevator
        //can return floors 1 to 5
        //if returns -1, the elevator is not on any floor
        public int whereIs()
        {
            //gets the byte output
            byte output = c.read(c.port3in);

            //return floor numbers
            if (EightBit.getAt(output, 0) == 0)
                return 1;
            else if (EightBit.getAt(output, 1) == 0)
                return 2;
            else if (EightBit.getAt(output, 2) == 0)
```

```
                return 3;
            else if (EightBit.getAt(output, 3) == 0)
                return 4;
            else if (EightBit.getAt(output, 4) == 0)
                return 5;
            else
                return -1;
        }

        //returns true if door is open
        public bool isDoorOpen()
        {
            //gets the byte output
            byte output = c.read(c.port4in);

            //return floor numbers
            if (EightBit.getAt(output, 7) == 0)
                return false;
            else
                return true;
        }

        //returns true if floor sensor is on
        public bool isFloorSensorOn()
        {
            //gets the byte output
            byte output = c.read(c.port4in);

            //return floor numbers
            if (EightBit.getAt(output, 6) == 0)
                return true;
            else
                return false;
        }
    }
}
```

ElevatorController.cs

```
using System;
using System.Threading;

namespace Herko.Elevator
{
    internal class ElevatorController
    {
        public Elevator elevator;

        public bool isGoing = false;
        public bool prevDoorStatus = false;
        public bool prevFloorStatus = false;

        //ACTIVE LOG.0!!!!!
        public ElevatorController(Elevator e)
        {
            //default
            this.elevator = e;
            elevator.reset();
            prevDoorStatus = elevator.isDoorOpen();
            prevFloorStatus = elevator.isFloorSensorOn();
        }

        //makes elevator go up
        public void goUp()
```

```
        {
            elevator.engineUp();
            elevator.engineOn();

            elevator.ledUpOn();
            elevator.ledDownOff();

            isGoing = true;

            Thread.Sleep(300);
        }

        //makes elevator go down
        public void goDown()
        {
            elevator.engineDown();
            elevator.engineOn();

            elevator.ledUpOff();
            elevator.ledDownOn();

            isGoing = true;

            Thread.Sleep(300);
        }

        //stops the elevator
        public void stop()
        {
            elevator.engineOff();

            elevator.ledUpOff();
            elevator.ledDownOff();

            elevator.playSound(100);

            isGoing = false;
        }

        //gets floor number
        public int getCurrentFloor()
        {
            return elevator.whereIs();
        }

        //returns true if elevator is on a floor
        public bool isOnFloor()
        {
            int pos = elevator.whereIs();

            if (pos == -1)
                return false;
            else
                return true;
        }

        //goes to floor [floor]
        public void goToFloor(int targetFloor)
        {
            int currentFloor = getCurrentFloor();

            if (targetFloor > currentFloor)
                goUp();
            else if (targetFloor < currentFloor)
```

```
                goDown();

            while (true)
            {
                if (getCurrentFloor() == targetFloor)
                {
                    stop();
                    break;
                }
            }
        }

        //goes to random floor to fix stuffs
        public void recall()
        {
            Console.WriteLine("[ELEVATOR] recalling...");

            goUp();
            int i = 0;
            while (i < 1000)
            {
                if (getCurrentFloor() == -1)
                {
                    Thread.Sleep(10);
                    i++;
                }
                else
                {
                    stop();
                    break;
                }
            }
        }

        //returns true if door status changed
        public bool doorChanged()
        {
            bool output = elevator.isDoorOpen();
            if (prevDoorStatus != output)
            {
                Console.WriteLine($"[ELEVATOR] door opened status {output}");
                prevDoorStatus = output;
                return true;
            }
            return false;
        }

        //returns true if floor sensor status changed
        public bool floorSensorChanged()
        {
            bool output = elevator.isFloorSensorOn();
            if (prevFloorStatus != output)
            {
                Console.WriteLine($"[ELEVATOR] floor sensor status {output}");
                prevFloorStatus = output;
                return true;
            }
            return false;
        }
    }
}
```

Buttons.cs

```
using Dejvoss.BitShift;
using System;

namespace Herko.Elevator
{
    internal class Buttons
    {
        //board controller
        public Controller c;

        public Buttons(Controller c)
        {
            this.c = c;
        }

        //returns the number of floor to go to after any button is pressed
        public int getFloorButtons()
        {
            int num = floorButtonPressed();
            if (num != -1)
            {
                Console.WriteLine($"[ELEVATOR] pressed floor button to floor {num}");
                return num;
            }

            num = cabinButtonPressed();
            if (num != -1)
            {
                Console.WriteLine($"[ELEVATOR] pressed cabin button to floor {num}");
                return num;
            }

            return -1;
        }

        //returns floor number at which a button got pressed
        //can return floors 1 to 5
        //if returns -1, no buttons are pressed
        public int floorButtonPressed()
        {
            //gets the byte output
            byte output = c.read(c.port4in);

            //return floor numbers
            if (EightBit.getAt(output, 0) == 0)
                return 1;
            else if (EightBit.getAt(output, 1) == 0)
                return 2;
            else if (EightBit.getAt(output, 2) == 0)
                return 3;
            else if (EightBit.getAt(output, 3) == 0)
                return 4;
            else
                return -1;
        }


        //returns floor number to which a button in the cabin got pressed
        //can return floors 1 to 5
        //if returns -1, no buttons are pressed
        public int cabinButtonPressed()
        {
            //gets the byte output
```

```
            byte p3output = c.read(c.port3in);
            byte p4output = c.read(c.port4in);

            //return floor numbers
            if (EightBit.getAt(p3output, 5) == 0)
                return 1;
            else if (EightBit.getAt(p3output, 6) == 0)
                return 2;
            else if (EightBit.getAt(p3output, 7) == 0)
                return 3;
            else if (EightBit.getAt(p4output, 5) == 0)
                return 4;
            else
                return -1;
        }
    }
}
```

Odpovědi na otázky

1) V průmyslových zařízeních je často nutné přenášet informaci mezi jednotlivými částmi. Jedním z protokolů vhodných pro tento účel je CANOpen. Zkuste popsat jeho základní principy a rámcově specifikovat velikost přenášené informace.

> Je to velmi sofistikovaný a složitý protokol, používá se například u servomotorů a jiných robotických zařízení. Používá sedmi vrstvý OSI model. Komunikaci začne vysláním 11 bitového CAN ID, a následně pošle 64 bitů s daty pro koncové zařízení.

2) Pro výtah ve vysoké budově s desítkami pater by asi nebylo nejpohodlnější (a také nejlevnější) natahovat vedení od každého tlačítka na patře do centrální řídící jednotky. Pokuste se navrhnout vhodnější řešení tohoto problému.

> Tady by možná stačil jeden vodič pro celou budovu, natažený z nejvyššího patra až do ovládací jednotky, ve kterém by se sériově posílal signál ze zmáčknutého tlačítka na patře.

3) Zkuste vymyslet způsob, jakým byste zjišťovali rozmístění vývodů LED displeje neznámého typu, tj. který vývod ovládá jaký segment displeje. K dispozici máte základní elektronické vybavení – multimetr, sadu odporů, napájecí zdroj.

> Nemám nejmenší tušení.

Závěr

Program sice funguje, ale nestihl jsem ho plně dokončit. Základní vlastnosti v programu jsou, kromě zabezpečení výtahu při pohybu. Výtah jezdí mezi patry a rozsvěcí se. Fungují jak tlačítka na patrech tak ty v kabině. Podlahové čidlo v sobě má nějaký bug který jsem nestihl opravit a segmentový displej neukazuje číslo patra.