PRAXE

č. 4	STŘEDNÍ PRŮMYSLOVÁ ŠKOLA CHOMUTOV	David Herko
25. 2. 2025	PŘÍSTUPOVÝ SYSTÉM S RFID	V4

Zadání

S využitím periferie čteček RFID čipů připojené pomocí USB / sériového rozhraní sestavte aplikaci, která bude na základě informací uložených v SQLite databázi vyhodnocovat oprávněnost požadavku na přístup z daného místa. Výsledné řešení musí obsahovat následující funkcionalitu:

- Obsluha dvou čteček připojených na periferii pomocí RS485 rozhraní

- Identifickace čtečky a způsobu načtení informace (načtení čipu, manuální zadání kódu na klávesnici). Simulujte tři místa události (klávesnice u vstupu, čip v místě A, čip v místě B)

- Identifickace osoby a vyhodnocení oprávněnosti přístupu na základě místa události, aktuálního času a dne v týdnu

- Uživatelsky použitelné rozhraní pro nastavení údajů do databáze (vlastník, členství ve skupinách, nastavení přístupových oprávnění a pravidel pro přístup)

- Zobrazení výsledku vyhodnocení žádosti a logování všech událostí do tabulky databáze ve formátu umožňujícím efektivní prohledávání na úrovni SQL dotazu (zamyslete se jak by mohlo vypadat např. dohledání vstupů do konkrétního prostoru)

Teorie

RFID (Radio-frequency identification) používá rádiovou frekvenci pro identifikaci čipů, které se mohou používat například pro zboží v obchodech nebo jako identifikace zaměstnanců. RFID systém se k počítači připojí pomocí sériové linky.

Po naskenování čipu po sériové lince přijde následující blok dat:

<START>	[ADDRESS]	ASCII TAG	CR	LF	<END>
1 bit	1 bit, nepovinné		1 bit	1 bit	1 bit

Podle adresy se určí, z jaké čtečky signál přišel; ASCII TAG je samotný obsah tagu (identifikátor).

Popis řešení

Po spuštění programu se otevře port na sériové lince a čeká na příchozí data. Pokud neexistuje databáze, vytvoří se i s (prázdnými) tabulkami. Po přijetí informací na sériové lince se vytvoří nový RFIDTag objekt, který načte informace z přijatého stringu. Podle lokace TAGu se určí dveře, u kterých si zaměstnanec ‚pípnul'. Z databáze se načte, jestli tam zaměstnanec má mít přístup, jestli ne, nepustí ho to. To samé se provede pro skupinu, ve které zaměstnanec je. Dále se zkontroluje čas, kdy zaměstnanec může přijít do práce. Pokud při jakémkoliv kroce něco selže, zaměstnanec není vpuštěn. Pokud vše proběhne v pořádku, zaměstnanec je vpuštěn a do databáze příchodů je přidán nový záznam.

Pokud chce administrátor přidat nového uživatele, může použít menu nahoře programu, to samé platí pro přidávání uživatelských skupin. Obě tyhle možnosti mají (docela) přívětivé uživatelské rozhraní. Po zmáčknutí tlačítek pro přidání uživatele či skupiny program zjistí zadané informace a přidá je do příslušných databází.

S databází pracuji přes knihovnu `Microsoft.Data.Sqlite`, kterou volám ve vlastní classe Database.cs pro jednodužší práci. Schéma databáze je následující:

TABLE entries – ukládá všechny pokusy o načtení tagů

`INT id`	`TEXT user`	`TEXT location`	`INT time`	`TEXT error`
ID příchodu	Jméno uživatele	Lokace čtečky	UNIX čas načtení	Chybová hláška

TABLE users – ukládá informace o zaměstnancích

INT id	TEXT tag_id	TEXT user	INT gate_pin	INT access
ID uživatele	Obsah/ID tagu	Jméno uživatele	PIN pro vstup	ID přístupové skupiny

TABLE groups – ukládá informace o zaměstnaneckých skupinách

INT id	TEXT name	INT access_gate	INTEGER access_room1	INTEGER access_room2	
ID skupiny	Jméno skupiny	Přístup k hlavní bráně	Přístup do místnosti 1	Přístup do místnosti 2	
TEXT time_monday	TEXT time_tuesday	TEXT time_wednesday	TEXT time_thursday	TEXT time_friday	TEXT time_weekends
Čas přístupu v pondělí	Čas přístupu v úterý	Čas přístupu ve středu	Čas přístupu ve čtvrtek	Čas přístupu v pátek	Čas přístupu o víkendech

Rozbor proměnných a metod

Form1.cs – hlavní GUI okno, práce s daty a sériovou linkou

public static SerialPort comPort	Port sériové linky
public static Database db	Odkaz na databázi
public DayOfWeek[] days	Dny v týdnu
public String[] daysStr	Dny v týdnu jako stringové hodnoty
public String[] roomsStr	Dostupné místnosti jako stringové hodnoty
public Form1()	Konstruktor
private void comPort_DataReceived(..)	Data receive sériové linky
private void reloadEntries()	Načítá informace o vstupech do místností
private void reloadUsers()	Načítá informace o zaměstnancích
private void addUserToolStripMenuItem_Click(..)	Zmáčknutí tlačítka pro přidání zaměstnance
private void addUsergroupToolStripMenuItem_Click(..)	Zmáčknutí tlačítka pro přidání skupiny
private void aboutToolStripMenuItem_Click(..)	Zobrazí info o programu

Database.cs – práce s databází

public SqliteConnection connection	Připojení k databázi
public Database(String databaseName)	Konstruktor – vytváří sqliteconnection
public void runCommand(String commandString)	Spustí SQLite příkaz

`public object selectCommand(String commandString)`	Spustí SQLite SELECT příkaz a vrátí hodnotu
`public List<String[]> selectCommandList(String commandString)`	Spustí SQLite SELECT příkaz a vrátí hodnoty jako List

NewUser.cs – GUI, přidávání nových zaměstnanců

`public SerialPort comPort`	Sériová linka
`public NewUser()`	Konstruktor
`private void btnAdd_Click(..)`	„Add user“ tlačítko, přidá zaměstnance se zadanými hodnotami
`private void comPort_DataReceived(..)`	Data receive sériové linky, pro načítání TAGů
`private void NewUser_Load(..)`	Po načtení okna

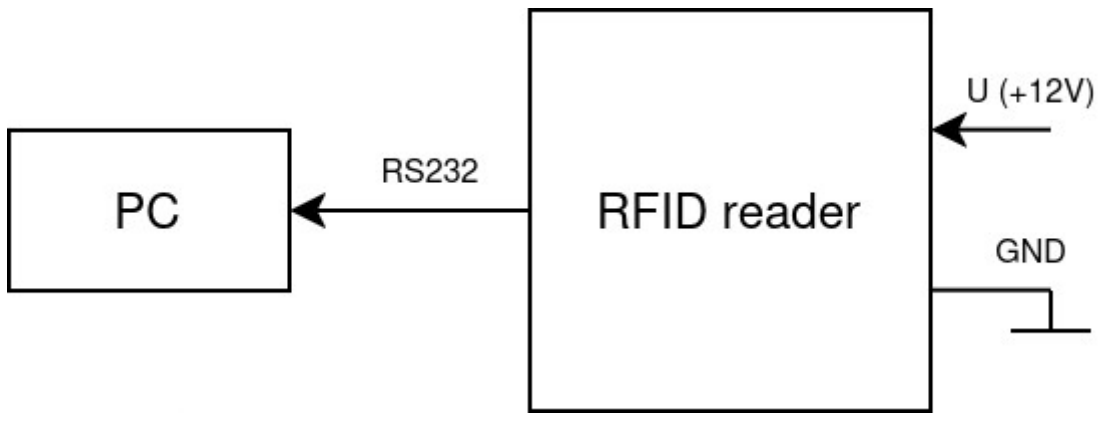
NewGroup.cs – GUI, přidávání nových skupin

`public NewGroup()`	Konstruktor
`private void chkMonday_CheckedChanged(..)`	Je-li checkBox pro pondělí zaškrtnut
`… to samé pro úterý až víkendy`	Je-li checkBox pro [den] zaškrtnut
`private void btnAdd_Click(..)`	„Add usergroup“ tlačítko, přidá skupinu se zadanými hodnotami, pokud jsou validní

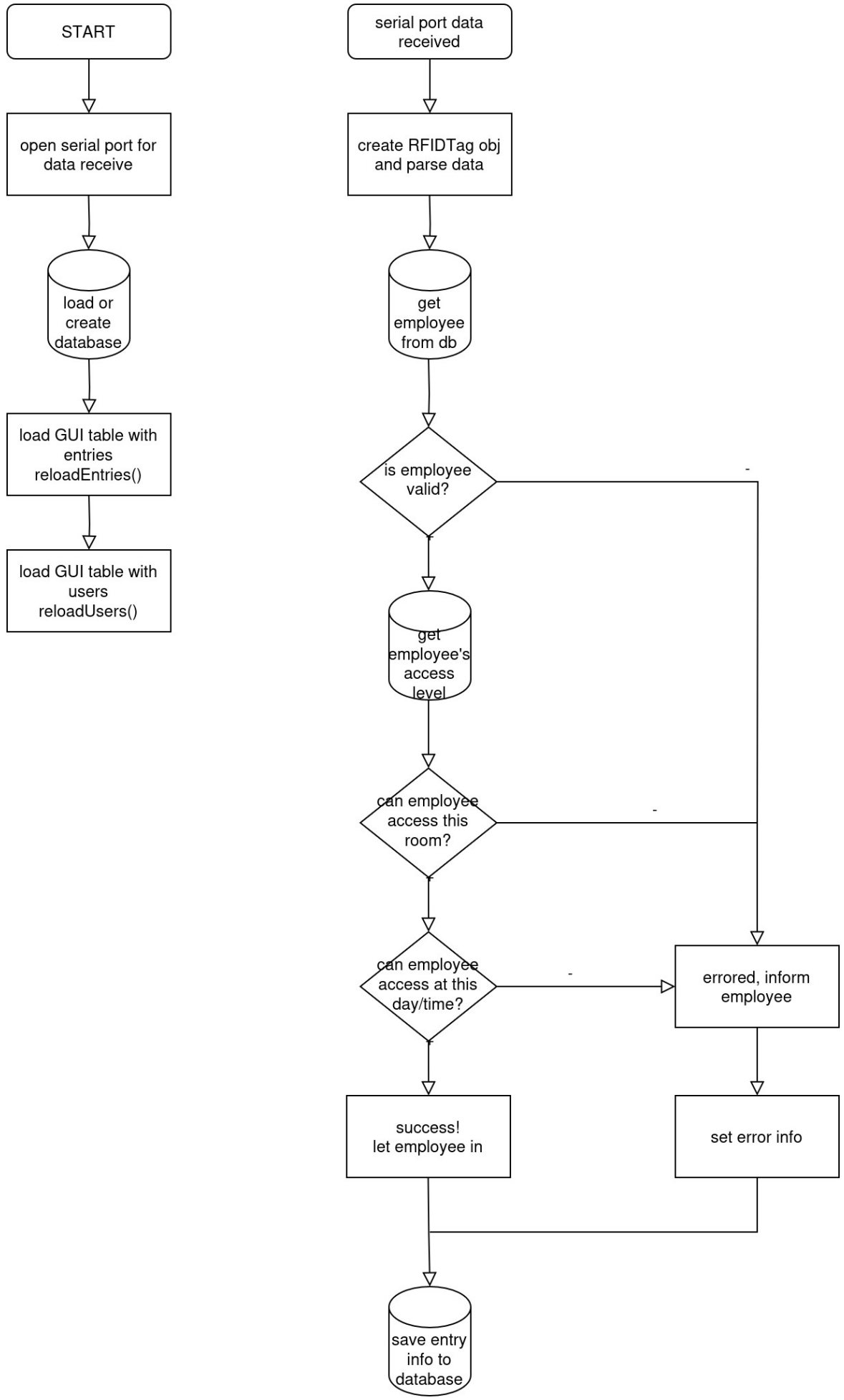
RFIDTag.cs – objekt pro ukládání hodnot TAGu

`public String location`	Odkud byl TAG načten
`public String data`	Co bylo v TAGu (ID uživatele)
`public String errorMsg`	Chybová hláška pro vrácení
`public RFIDTag(String readStr)`	Konstruktor

Schéma zapojení



Vývojový diagram



Komentovaný výpis kódu

Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Diagnostics;
using System.IO.Ports;
using System.Windows.Forms;

namespace Herko.RFID
{
    public partial class Form1 : Form
    {
        public static SerialPort comPort;
        public static Database db;

        public DayOfWeek[] days = new DayOfWeek[] { DayOfWeek.Monday,
DayOfWeek.Tuesday, DayOfWeek.Wednesday, DayOfWeek.Thursday, DayOfWeek.Friday,
DayOfWeek.Saturday, DayOfWeek.Sunday };
        public String[] daysStr = new String[] { "monday", "tuesday", "wednesday",
"thursday", "friday", "weekend", "weekend" };
        public String[] roomsStr = new String[] { "gate", "room1", "room2" };

        public Form1()
        {
            InitializeComponent();
            this.Text = "Herko.RFID | (c) DEJVOSS Productions 2025";
            this.MinimizeBox = false;
            this.MaximizeBox = false;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            DataGridView.CheckForIllegalCrossThreadCalls = false;

            comPort = new SerialPort("COM3", 19200, Parity.None, 8, StopBits.One);

            comPort.Open();
            comPort.DataReceived += comPort_DataReceived;

            db = new Database("rfid.db");
            db.runCommand("CREATE TABLE IF NOT EXISTS users (" +
                    "id INTEGER UNIQUE PRIMARY KEY AUTOINCREMENT," +
                    "tag_id TEXT UNIQUE NOT NULL," +
                    "user TEXT UNIQUE NOT NULL," +
                    "gate_pin INTEGER NOT NULL," +
                    "access INTEGER NOT NULL" +
                    ");");

            db.runCommand("CREATE TABLE IF NOT EXISTS entries (" +
                    "id INTEGER UNIQUE PRIMARY KEY AUTOINCREMENT," +
                    "user TEXT NOT NULL," +
                    "location TEXT NOT NULL," +
                    "time INTEGER NOT NULL," +
                    "error TEXT" +
                    ");");

            db.runCommand("CREATE TABLE IF NOT EXISTS groups (" +
```

```
                    "id INTEGER UNIQUE PRIMARY KEY AUTOINCREMENT," +
                    "name TEXT NOT NULL," +
                    "access_gate INTEGER NOT NULL," +
                    "access_room1 INTEGER NOT NULL," +
                    "access_room2 INTEGER NOT NULL," +
                    "time_monday TEXT," +
                    "time_tuesday TEXT," +
                    "time_wednesday TEXT," +
                    "time_thursday TEXT," +
                    "time_friday TEXT," +
                    "time_weekend TEXT" +
                    ");");

            reloadEntries();
            dataViewEntries.Columns[0].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dataViewEntries.Columns[1].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dataViewEntries.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dataViewEntries.Columns[3].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dataViewEntries.Columns[4].AutoSizeMode =
DataGridViewAutoSizeColumnMode.Fill;

            reloadUsers();
        }


        private void comPort_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
            Debug.WriteLine("read somethig!!!");

            //read from the com port
            String read = comPort.ReadExisting();
            Debug.WriteLine(read);
            Debug.WriteLine(read.Length);

            //create a tag object which parses data
            RFIDTag tag = new RFIDTag(read);
            if (tag.data == null) //if it returns null then an error occured
            {
                MessageBox.Show("Whoops! Something went wrong!");
                return;
            }

            //build command
            String username = null;
            Int64 accessLvl = 0;
            String errorMsg = tag.errorMsg;

            //get username
            if (tag.location == "gate")
                username = (String)db.selectCommand($"SELECT user from users WHERE
gate_pin = \"{tag.data}\";");
            else
                username = (String)db.selectCommand($"SELECT user from users WHERE
tag_id = \"{tag.data}\";");
```

```
            if (String.IsNullOrWhiteSpace(username))
            {
                errorMsg = "Unknown user";
                goto FINISH;
            }

            //get access level
            if (tag.location == "gate")
                accessLvl = (Int64)db.selectCommand($"SELECT access from users WHERE
gate_pin = \"{tag.data}\";");
            else
                accessLvl = (Int64)db.selectCommand($"SELECT access from users WHERE
tag_id = \"{tag.data}\";");

            Debug.WriteLine(username + "!!!!");

            //get current date+time
            DateTime dt = DateTime.Now;

            List<String[]> entriesData = db.selectCommandList($"SELECT * from groups");
            String[] permissions = null;
            for (int x = 0; x < entriesData.Count; x++)
            {
                if (int.Parse(entriesData[x][0]) == accessLvl)
                {
                    permissions = entriesData[x];
                    break;
                }
            }

            if (permissions == null)
            {
                errorMsg = "Invalid group";
                goto FINISH;
            }

            //block entry depending on room
        int roomIndex = Array.IndexOf(roomsStr, tag.location);
            Debug.WriteLine("ROOM " + permissions[2 + roomIndex]); //2 + roomIndex gets
correct room we want
            int allowed = int.Parse(permissions[2 + roomIndex]);
            if (allowed != 1)
            {
                errorMsg = "Entry not allowed!";
                goto FINISH;
            }

            //block entry depending on day or time
            int index = Array.IndexOf(days, dt.DayOfWeek);
            Debug.WriteLine("DAY " + daysStr[index]);
            Debug.WriteLine(permissions[1].ToString());
            String allowedEntry = permissions[5 + index].ToString(); //5 + index gets
the correct day we want

            if (allowedEntry != "NULL")
            {
                String[] split = allowedEntry.Split(';');
```

```
                DateTime min = DateTime.Parse(split[0]);
                DateTime max = DateTime.Parse(split[1]);

                Debug.WriteLine("min " + min.TimeOfDay.ToString());
                Debug.WriteLine("max " + max.TimeOfDay.ToString());

                if (dt.TimeOfDay < min.TimeOfDay)
                    errorMsg = "Unallowed time";
                else if (dt.TimeOfDay > max.TimeOfDay)
                    errorMsg = "Unallowed time";
            }
            else
            {
                errorMsg = "Unallowed time";
            }

        FINISH:
            dt = DateTime.Now;
            Int32 unix = (int)dt.Subtract(new DateTime(1970, 1, 1)).TotalSeconds;

            Debug.WriteLine($"{username} beeped at {tag.location} @ {dt.Hour}:
{dt.Minute}:{dt.Second} on {dt.Day}.{dt.Month}.{dt.Year} ({unix})");

            //add entry info to db
            db.runCommand($"INSERT INTO entries (\'user\', \'location\', \'time\',
\'error\') VALUES(\'{username}\', \'{tag.location}\', {unix}, \'{errorMsg}\');");

            reloadEntries();
        }

        //reloads entries from database
        private void reloadEntries()
        {
            dataViewEntries.Rows.Clear();
            dataViewEntries.Columns.Clear();

            dataViewEntries.Columns.Add("id", "ID");
            dataViewEntries.Columns.Add("user", "Employee");
            dataViewEntries.Columns.Add("location", "Location");
            dataViewEntries.Columns.Add("time", "Time");
            dataViewEntries.Columns.Add("error", "Error");

            List<String[]> entriesData = db.selectCommandList($"SELECT * FROM
entries");
            for (int y = 0; y < entriesData.Count; y++)
            {
                String[] row = entriesData[y];

                //parse time to a readable format
                DateTime dt = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Local);
                dt = dt.AddSeconds(int.Parse(row[3])).ToLocalTime();
                row[3] = dt.ToString("HH:mm:ss dd.MM.yyyy");

                dataViewEntries.Rows.Insert(y, row);
            }

            dataViewEntries.Sort(dataViewEntries.Columns["time"],
ListSortDirection.Descending);
```

```
        }

        //reloads users from database
        private void reloadUsers()
        {
            dataViewUsers.Rows.Clear();
            dataViewUsers.Columns.Clear();

            dataViewUsers.Columns.Add("id", "ID");
            dataViewUsers.Columns.Add("tag_id", "Tag ID");
            dataViewUsers.Columns.Add("user", "Employee");
            dataViewUsers.Columns.Add("gate_pin", "Gate PIN");
            dataViewUsers.Columns.Add("access", "User group");

            List<String[]> entriesData = db.selectCommandList($"SELECT * FROM users;");
            for (int y = 0; y < entriesData.Count; y++)
            {
                String[] row = entriesData[y];
                dataViewUsers.Rows.Insert(y, row);
            }

            dataViewUsers.Sort(dataViewUsers.Columns["id"],
ListSortDirection.Ascending);
            dataViewUsers.Columns["id"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
        }

        private void addUserToolStripMenuItem_Click(object sender, EventArgs e)
        {
            comPort.DataReceived -= comPort_DataReceived;

            NewUser nu = new NewUser();
            nu.ShowDialog();

            comPort.DataReceived += comPort_DataReceived;
            reloadUsers();
        }

        private void addUsergroupToolStripMenuItem_Click(object sender, EventArgs e)
        {
            NewGroup ng = new NewGroup();
            ng.ShowDialog();
        }

        private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Herko.RFID\n(c) DEJVOSS Productions 2025\nAn RFID system
made as a school project", "About | Herko.RFID", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
    }
}
```

Database.cs

```
using Microsoft.Data.Sqlite;
using System;
```

```
using System.Collections.Generic;

namespace Herko.RFID
{
    //DbWorker - wrapper for Microsoft.Data.Sqlite;
    //(c) DEJVOSS Productions 2025;
    public class Database
    {
        public SqliteConnection connection = null;

        public Database(String databaseName)
        {
            connection = new SqliteConnection($"Data source={databaseName}");
            connection.Open();
        }

        public void runCommand(String commandString)
        {
            if (connection == null)
                throw new Exception("Not connected to a dabase!");

            SqliteCommand command = connection.CreateCommand();
            command.CommandText = commandString;
            command.ExecuteNonQuery();
        }

        public object selectCommand(String commandString)
        {
            if (connection == null)
                throw new Exception("Not connected to a dabase!");

            SqliteCommand command = connection.CreateCommand();
            command.CommandText = commandString;
            return command.ExecuteScalar();
        }

        //returls whole table as a list
        public List<String[]> selectCommandList(String commandString)
        {
            if (connection == null)
                throw new Exception("Not connected to a dabase!");

            SqliteCommand command = connection.CreateCommand();
            command.CommandText = commandString;
            command.CommandType = System.Data.CommandType.Text;

            SqliteDataReader r = command.ExecuteReader();
            List<String[]> final = new List<String[]>();

            while (r.Read())
            {
                String[] oneRow = new String[r.FieldCount];

                for (int i = 0; i < r.FieldCount; i++)
                {
                    oneRow[i] = $"{r[i]}";
                    //Debug.WriteLine("TEST" + r[i]);
                }
```

```
                final.Add(oneRow);
            }

            return final;
        }
    }
}
```

NewUser.cs

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO.Ports;
using System.Windows.Forms;

namespace Herko.RFID
{
    public partial class NewUser : Form
    {
        public SerialPort comPort;

        public NewUser()
        {
            InitializeComponent();
            this.MinimizeBox = false;
            this.MaximizeBox = false;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;

            comPort = Form1.comPort;
            comPort.DataReceived += comPort_DataReceived;

            //load the usergroups
            List<String[]> entriesData = Form1.db.selectCommandList($"SELECT * FROM
groups");
            for (int y = 0; y < entriesData.Count; y++)
            {
                String[] row = entriesData[y];
                comboAccess.Items.Add($"{row[0]} | {row[1]}");
            }
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            //build sqlite cmd
            String cmd = $"INSERT INTO users (\'tag_id\', \'user\', \'gate_pin\',
\'access\') VALUES(\'{boxTagId.Text}\', \'{boxUsername.Text}\',
{numPin.Value.ToString()}, {comboAccess.SelectedIndex + 1});";

            //run and close
            Form1.db.runCommand(cmd);
            comPort.DataReceived -= comPort_DataReceived;
            this.Close();
        }

        private void comPort_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
```

```
            Debug.WriteLine("read somethig!!!");

            //read from the com port
            String read = comPort.ReadExisting();
            Debug.WriteLine(read);
            Debug.WriteLine(read.Length);

            //create a tag object which parses data
            RFIDTag tag = new RFIDTag(read);
            if (tag.data == null) //if it returns null then an error occured
            {
                MessageBox.Show("Whoops! Something went wrong!");
                return;
            }

            //add tag id text to box
            if (tag.location != "gate")
            {
                boxTagId.Text = tag.data;
            }
        }

        private void NewUser_Load(object sender, EventArgs e)
        {
            if (comboAccess.Items.Count > 0)
                comboAccess.SelectedIndex = 0;
            else
            {
                MessageBox.Show("No usergroup exists!\nCreate one first.", "Error!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                comPort.DataReceived -= comPort_DataReceived;
                this.Close();
            }
        }
    }
}
```

NewGroup.cs

```
using System;
using System.Windows.Forms;

namespace Herko.RFID
{
    public partial class NewGroup : Form
    {
        public NewGroup()
        {
            InitializeComponent();
            this.MinimizeBox = false;
            this.MaximizeBox = false;
            this.FormBorderStyle = FormBorderStyle.FixedSingle;
        }

        private void chkMonday_CheckedChanged(object sender, EventArgs e)
        {
            if (chkMonday.Checked)
            {
```

```
            timeMonMax.Enabled = true;
            timeMonMin.Enabled = true;
        }
        else
        {
            timeMonMax.Enabled = false;
            timeMonMin.Enabled = false;
        }
    }

    private void chkTuesday_CheckedChanged(object sender, EventArgs e)
    {
        if (chkTuesday.Checked)
        {
            timeTueMax.Enabled = true;
            timeTueMin.Enabled = true;
        }
        else
        {
            timeTueMax.Enabled = false;
            timeTueMin.Enabled = false;
        }
    }

    private void chkWednesday_CheckedChanged(object sender, EventArgs e)
    {
        if (chkWednesday.Checked)
        {
            timeWedMax.Enabled = true;
            timeWedMin.Enabled = true;
        }
        else
        {
            timeWedMax.Enabled = false;
            timeWedMin.Enabled = false;
        }
    }

    private void chkThursday_CheckedChanged(object sender, EventArgs e)
    {
        if (chkThursday.Checked)
        {
            timeThuMax.Enabled = true;
            timeThuMin.Enabled = true;
        }
        else
        {
            timeThuMax.Enabled = false;
            timeThuMin.Enabled = false;
        }
    }

    private void chkFriday_CheckedChanged(object sender, EventArgs e)
    {
        if (chkFriday.Checked)
        {
            timeFriMax.Enabled = true;
            timeFriMin.Enabled = true;
```

```
            }
            else
            {
                timeFriMax.Enabled = false;
                timeFriMin.Enabled = false;
            }
        }

        private void chkWeekend_CheckedChanged(object sender, EventArgs e)
        {
            if (chkWeekend.Checked)
            {
                timeWeeMax.Enabled = true;
                timeWeeMin.Enabled = true;
            }
            else
            {
                timeWeeMax.Enabled = false;
                timeWeeMin.Enabled = false;
            }
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            //parse time data
            String mondayEntry = $"{timeMonMin.Text};{timeMonMax.Text}";
            if (!chkMonday.Checked)
                mondayEntry = "NULL";

            String tuesdayEntry = $"{timeTueMin.Text};{timeTueMax.Text}";
            if (!chkTuesday.Checked)
                tuesdayEntry = "NULL";

            String wednesdayEntry = $"{timeWedMin.Text};{timeWedMax.Text}";
            if (!chkWednesday.Checked)
                wednesdayEntry = "NULL";

            String thursdayEntry = $"{timeThuMin.Text};{timeThuMax.Text}";
            if (!chkThursday.Checked)
                thursdayEntry = "NULL";

            String fridayEntry = $"{timeFriMin.Text};{timeFriMax.Text}";
            if (!chkFriday.Checked)
                fridayEntry = "NULL";

            String weekendEntry = $"{timeWeeMin.Text};{timeWeeMax.Text}";
            if (!chkWeekend.Checked)
                weekendEntry = "NULL";

            //build sqlite cmd
            String cmd = $"INSERT INTO groups (\'name\', \'access_gate\',
\'access_room1\', \'access_room2\', " +
                $"\'time_monday\', \'time_tuesday\', \'time_wednesday\',
\'time_thursday\', \'time_friday\', \'time_weekend\') " +
                $"VALUES(\'{boxName.Text}\', {chkGate.Checked.ToString().ToUpper()},
{chkRoom1.Checked.ToString().ToUpper()}, {chkRoom2.Checked.ToString().ToUpper()}, " +
                $"\'{mondayEntry}\', \'{tuesdayEntry}\', \'{wednesdayEntry}\',
\'{thursdayEntry}\', \'{fridayEntry}\', \'{weekendEntry}\');";
```

```
            //run and close
            Form1.db.runCommand(cmd);
            this.Close();
        }
    }
}
```

RFIDTag.cs

```
using System;

namespace Herko.RFID
{
    internal class RFIDTag
    {
        public String location = null;
        public String data = null;
        public String errorMsg = null;

        public RFIDTag(String readStr)
        {
            //if entered pin at main gate
            if (readStr.Length == 9)
            {
                //get entered pin
                String hexa = $"{readStr[2]}{readStr[3]}{readStr[4]}{readStr[5]}";
                int enteredPin = int.Parse(hexa,
System.Globalization.NumberStyles.HexNumber);

                location = "gate";
                data = enteredPin.ToString();
                return;
            }

            //if invalid length
            if (readStr.Length != 13)
            {
                errorMsg = "Unknown location";
                return;
            }

            //if the code gets here it means all went correctly
            if (readStr[1] == 4) //if beeped at room1
            {
                location = "room1";
                for (int i = 2; i < 10; i++)
                {
                    data += readStr[i];
                }
            }
            else //if beeped at room2
            {
                location = "room2";
                for (int i = 1; i < 9; i++)
                {
                    data += readStr[i];
                }
```

```
            }
        }
    }
}
```

Odpovědi na otázky

1. Uveďte standardní frekvence, na kterých se používají RFID tagy (karty, přívěsky, etikety …). Jaký vliv má frekvence tagu na jeho praktickou použitelnost?

> Dělí se na tři skupiny – nízkofrekvenční (124 kHz nebo 135 kHz), vysokofrekvenční (13,56 MHz) a ultrafrekvenční (860 MHz do 960 MHz). Frekvence určuje, na jakou vzdálenost lze tag načíst.[1]

2. Z důvodu větší bezpečnosti se často v identifikačních systémech používají tzv. smart card. Najděte základní odlišnosti od běžného RFID tagu a vysvětlete jak je větší bezpečnosti identifikace při použití tohoto řešení dosaženo.

> Smart card v sobě ukládá více dat, například digitální certifikát, RFID tag může uložit jenom identifikátor. Smart card se také musí zasunout do čtečky, RFID tag stačí ke čtečce přiložit. Smart cards také umožňují šifrování a autentikaci pro získání dat. Všechny tyto vychytávky zvyšují bezpečnost.[2]

3. Pokuste se naznačit řešení spolehlivosti přístupového systému tak, aby výpadek řídící aplikace nebo centrálního počítače nezpůsobil kompletní znepřístupnění všech místností (např. v areálu firmy nebo hotelu).

> Asi by bylo nejlepší mít záložní generátor nebo staré dobré klíče.

Závěr

Toto zadání jsem poměrně stihl, moje řešení má sice nějaké ty háčky – chtělo by to kontrolu pro chybné hodnoty, občas se TAGy nestihnou načíst celé a GUI hlavního okna není nejlepší – ale program je plně funkční.

1 https://en.wikipedia.org/wiki/Radio-frequency_identification

2 https://www.as-rfid.com/info/difference-between-smart-card-rfid-card-81965460.html