SPRAWOZDANIE KOŃCOWE PT:

BUDOWA SYSTEMU KUCHENNEGO DLA OSÓB NIEPEŁNOSPRAWNYCH.

Opracowali:

Robert Sasal

Grzegorz Dygas

Tomasz Napora

Spis treści

Adaptacja systemu drzwi uchylnych

W ramach prac dokonano adaptacji zakupionego sytemu otwierania drzwi uchylnych. Adaptacji wymagał system bezpieczeństwa, oraz mocowanie siłownika i jego sterowanie.

Modyfikacji wysterowania dokonano za pomocą zmiany kodu aplikacji sterującej.

Natomiast modyfikacji technicznej dokonano podczas procesu montażu, co przedstawiają poniższe zdjęcia.

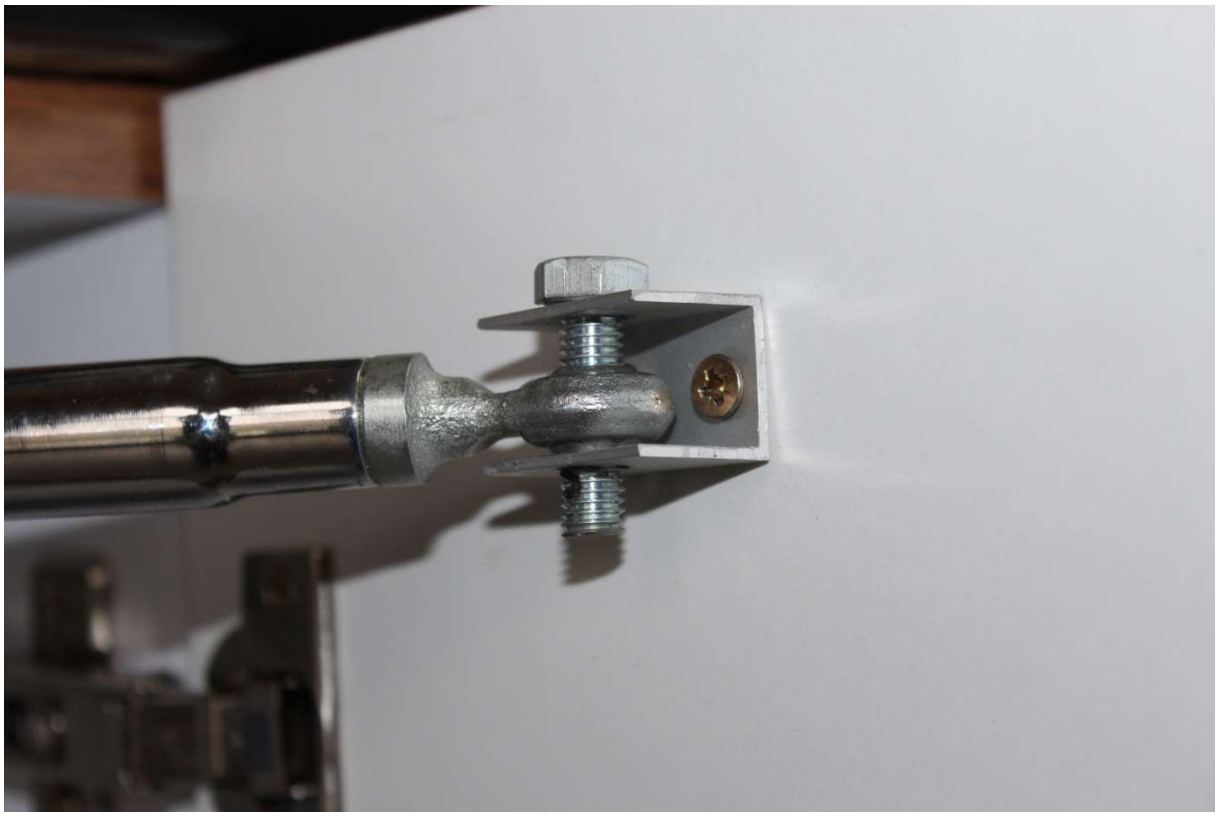
zdj 1. Mechanizm otwierania drzwi

zdj 2. System sterowania otwierania szafki wraz z mechanizmem zabezpieczającym i blokadą rodzicielską.

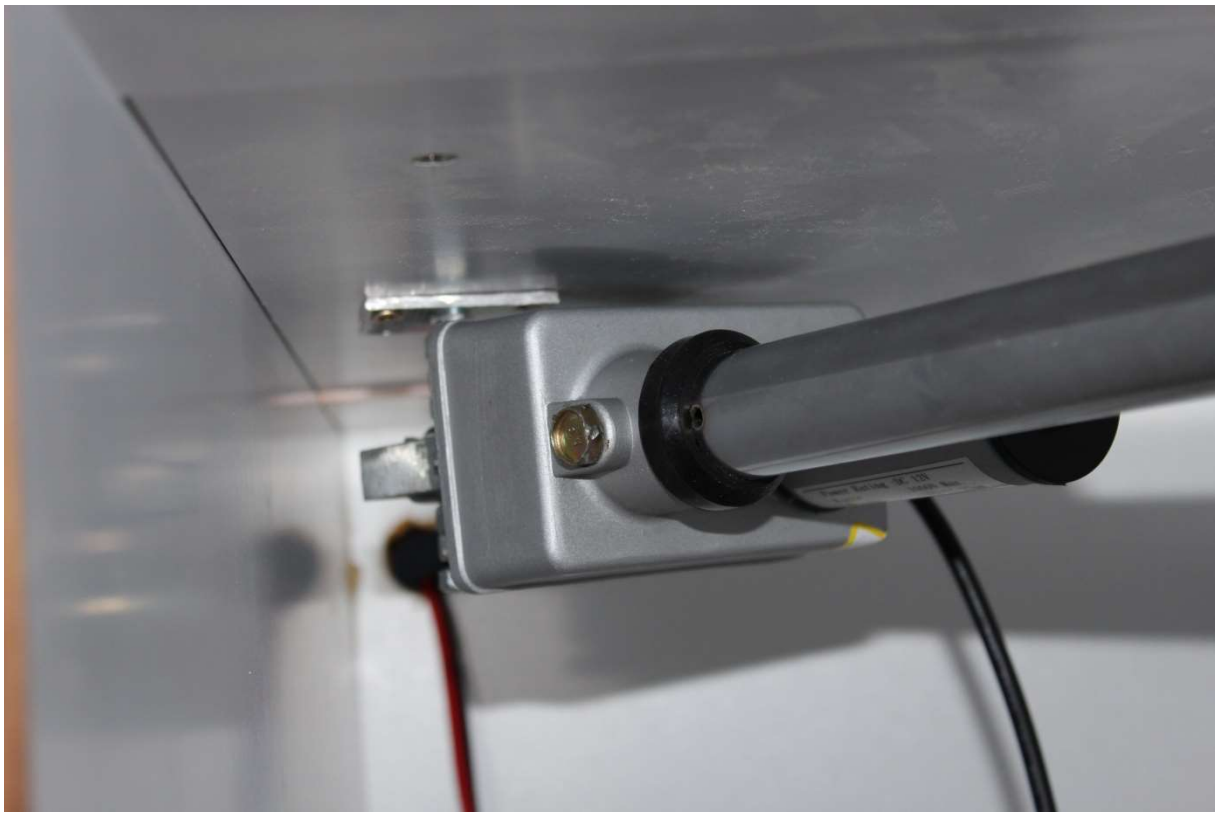
W ramach dodatkowej funkcyjności, w systemie otwierania drzwi wprowadzono funkcję blokady rodzicielskiej. Blokada ta pozwala zablokować z poziomu aplikacji możliwość otwarcia szafki od zewnątrz. Uniemożliwia tym samym dostęp do wnętrza szafki dzieciom lub osobom niepożądanym

zdj 3. Widok ogólny mechanizmu otwierania i zamykania szafki.

Siłownik zamocowano wahliwie do korpusu szafki oraz do drzwi zewnętrznych. W zabudowanym prototypie wykorzystano kątowniki aluminiowe, jako punkt obsadzenia wahliwych przyłączy. Podobnie jak przy rozwiązaniu wysuwania szuflady i znaczników krańcowego położenia, w przypadku seryjnego rozwiązania zakłada się zastąpienie elementów aluminiowych, dedykowanymi plastikowymi złączami. Poniższe zdjęcia ukazują punkty mocowania siłownika.

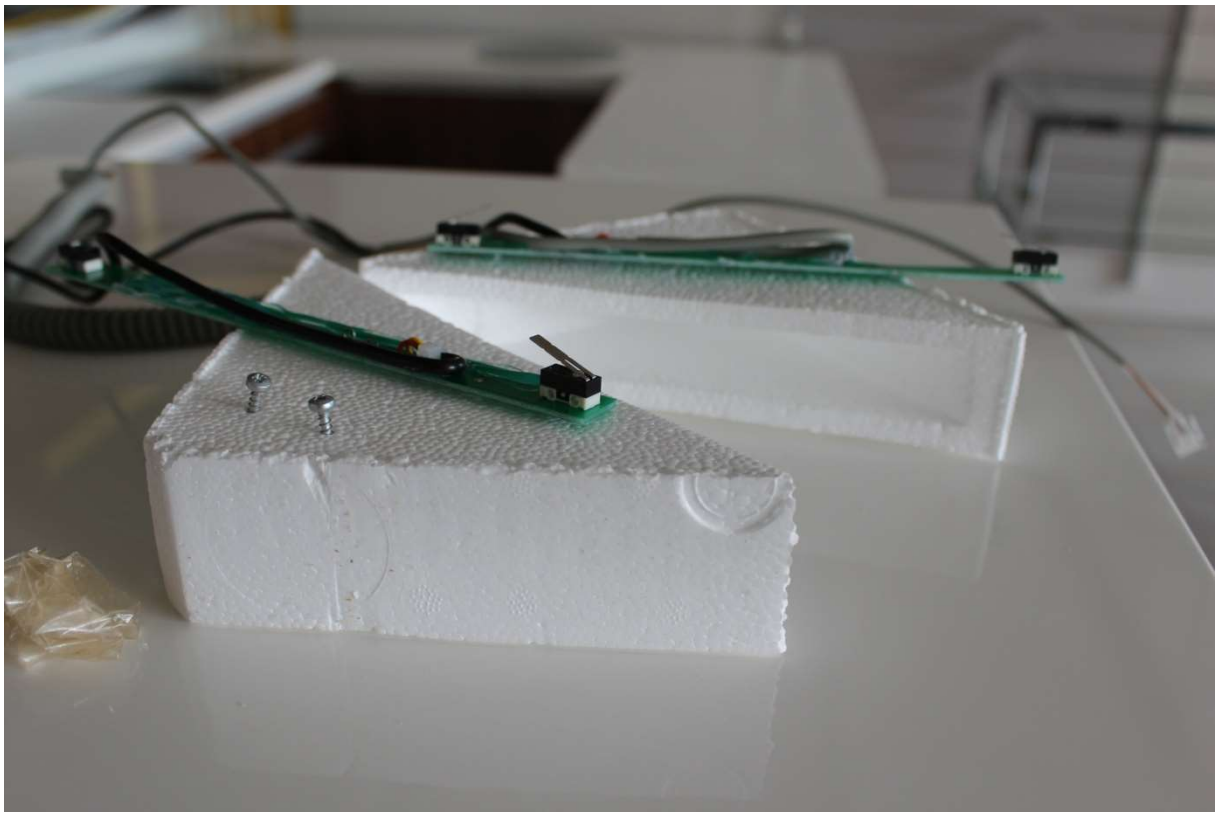


zdj 4. Mocowanie siłownika do drzwi



zdj 5. Mocowanie siłownika do korpusu szafki

Ponadto system zamykania drzwi szafki posiada zabezpieczenie przed przycięciem ręki. Do płyty stanowiącej drzwi zewnętrzne zamontowano ruchliwie dodatkową płytę z wyżłobionymi kanałami, w których ukryto styczniki. Zamocowanie płyty sprawia, że w razie natrafienia na przeszkodę podczas zamykania drzwi, wewnętrzna płyta ulega przesunięciu, a stycznik zamyka obwód bezpieczeństwa. Zamknięcie obwodu bezpieczeństwa powoduje wyłączenie systemu zamykania i jego odsunięcie o kilka milimetrów. Jest to sygnalizowane w aplikacji stosownym komunikatem. Na poniższym zdjęciu przedstawiono wyżłobiony kanał i czujnik stycznikowy.



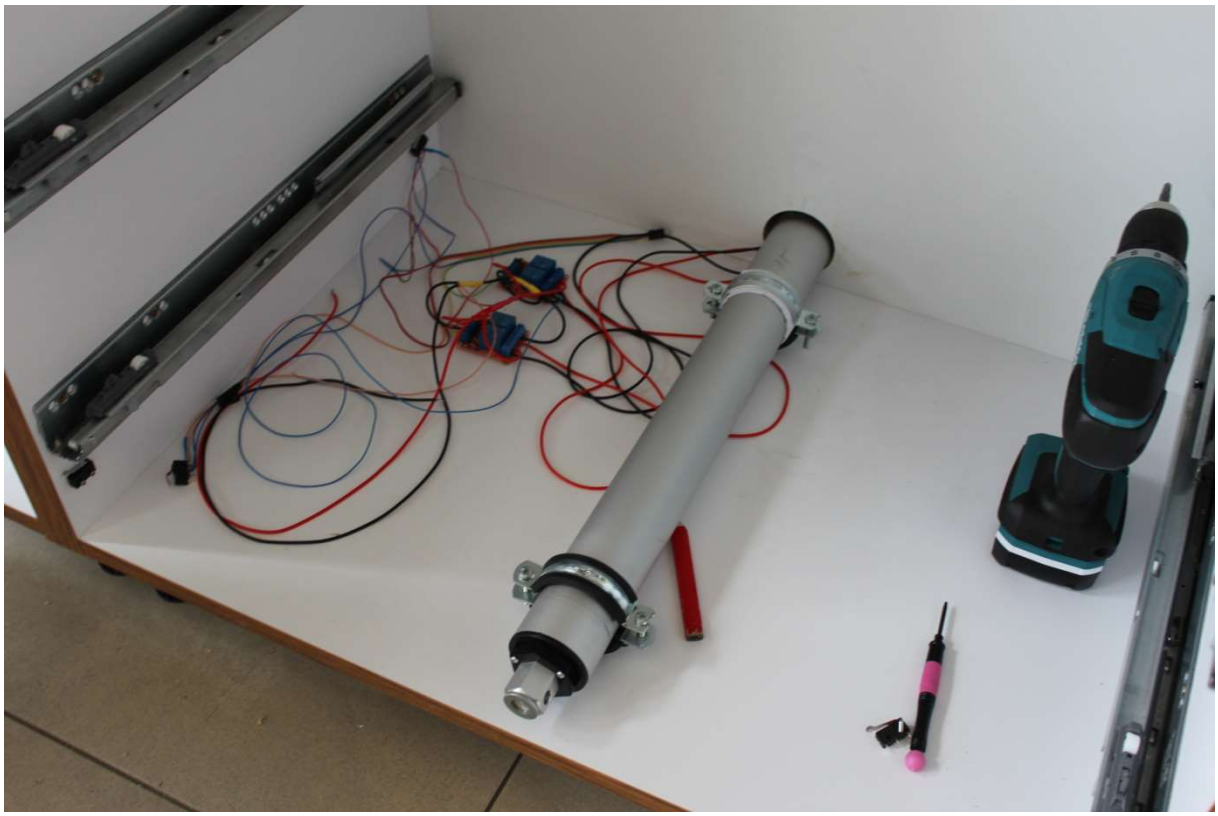
zdj 6. Styczniki do systemu bezpieczeństwa

zdj 7. Czujnik stycznikowy w płycie bezpieczeństwa

Adaptacja systemu odsuwania szuflad

W ramach prac dokonano adaptacji zakupionego sytemu odsuwania szuflad. W przeciwieństwie do rozwiązania zakupionego, zmodyfikowano umieszczenie siłownika. Ze względów praktyczno - estetycznych, zdecydowano o zamontowaniu napędu pod szufladą i wykonanie szuflady o płytszym dnie. Takie rozwiązanie sprawia, że system odsuwania szuflady jest kompletnie niewidoczny. Poniższe zdjęcia przedstawiają system podczas montażu.

zdj 8. Zamontowany siłownik, wraz z widocznymi prowadnicami szuflady



zdj 9. Sterowanie siłownika podczas montażu

Przy adaptacji systemu niezbędne było określenie długości na jaką wysuwa się szuflada. W tym celu zastosowano styczniki które po załączeniu przekazują do jednostki sterującej informację o krańcowych pozycjach szuflady. Takie rozwiązanie zapewnia dużą uniwersalność urządzenia i jedynym ograniczeniem jest długość na jaką wysuwa się siłownik. Planuje się wykonać odlewane plastikowe wskaźniki do styczników krańcowych. Na etapie montażu prototypowego rozwiązania zabudowy kuchennej, wykonano wskaźnik drewniany, widoczny na poniższym zdjęciu.

zdj 10. Wskaźnik do stycznika granicznego systemu odsuwania szuflady.

W ramach systemu bezpieczeństwa, zabezpieczeniem przed przycięciem palców podczas wsuwania szuflady, jest sposób mocowania siłownika do szuflady. Wykorzystano neodymowy magnes o dobranej doświadczalnie sile przyciągania. Szuflada z obciążeniem 10kg (maksymalna nośność prowadnic) zostaje wsunięta, natomiast w przypadku napotkania przeszkody, siłownik odłącza się od szuflady i ta ostatnia zostaje uwolniona.

Sterowanie systemem wysuwania szuflad połączono z aplikacją, co opisano w dalszej części sprawozdania.

Adaptacja systemu opuszczania szafek

W ramach prac dokonano adaptacji zakupionego od Politechniki Warszawskiej sytemu opuszczania. System został zabudowany w specjalnie do tego celu zbudowanej szafce meblarskiej. Stelaż szafki wykonano z tradycyjnych płyt wiórowych, natomiast drzwi i półki szafki wykonano z opracowanych w ramach niniejszych prac płyt meblarskich o ograniczonej masie. Poniższe zdjęcie przedstawia zabudowę stelaża w korpusie szafki meblarskiej.

zdj 11. Zabudowa stelaża podnoszenia/opuszczania w korpusie szafki meblarskiej.

Założeniem autorów konstrukcji stelaża była jego uniwersalność, umożliwiająca dostosowanie go w szerz (wymiana poprzeczek) jak i wzwyż (regulacja wysokości wysuwanych belek). Jest to rozwiązanie dobre w przypadku próby montażu stelaża do gotowej szafki. W naszym przypadku możliwe jest wykonanie szafki pod konkretne wymagania wymiarowe. W związku z powyższym dokonano modyfikacji mocowania, usuwając regulowane poprzeczki. Ich usunięcie umożliwiło spłycenie zabudowy stelaża, a co za tym idzie, większa powierzchnia wewnątrz szafki przeznaczona jest do wykorzystania. Zmodyfikowany aluminiowy stelaż szafki przedstawia poniższe zdjęcie.

zdj 12. Stelaż mechanizmu szafki opuszczanej z usuniętą poprzeczką regulacyjną.

Bardzo ważnym elementem w przypadku tego systemu jest układ zabezpieczający przed przygnieceniem opuszczaną szafką. Układ podobnie jak w przypadku otwieranych drzwi oparto na stycznikach, których wciśniecie wysyła informację do jednostki sterującej o występującej kolizji. Po zamknięciu obwodu system zatrzymuje opuszczanie szafki i cofa ją o kilka milimetrów zwalniając potencjalny ucisk. System następnie czeka na wydanie kolejnej komendy sterującej (szafka pozostaje w zawieszeniu). Styczniki zabudowano w płycie umieszczonej na sześciu kołkach - tj. dwóch na środku i czterech w narożnikach szafki. Płyta może się przemieszczać, a każde jej poruszenie powoduje uruchomienie stycznika. Poniższe zdjęcia pokazują płytę oraz pracę systemu zabezpieczającego.

zdj 13. Płyta ze stycznikami.

zdj 14. Płyta umieszczona w dolnej części szafki opuszczanej.

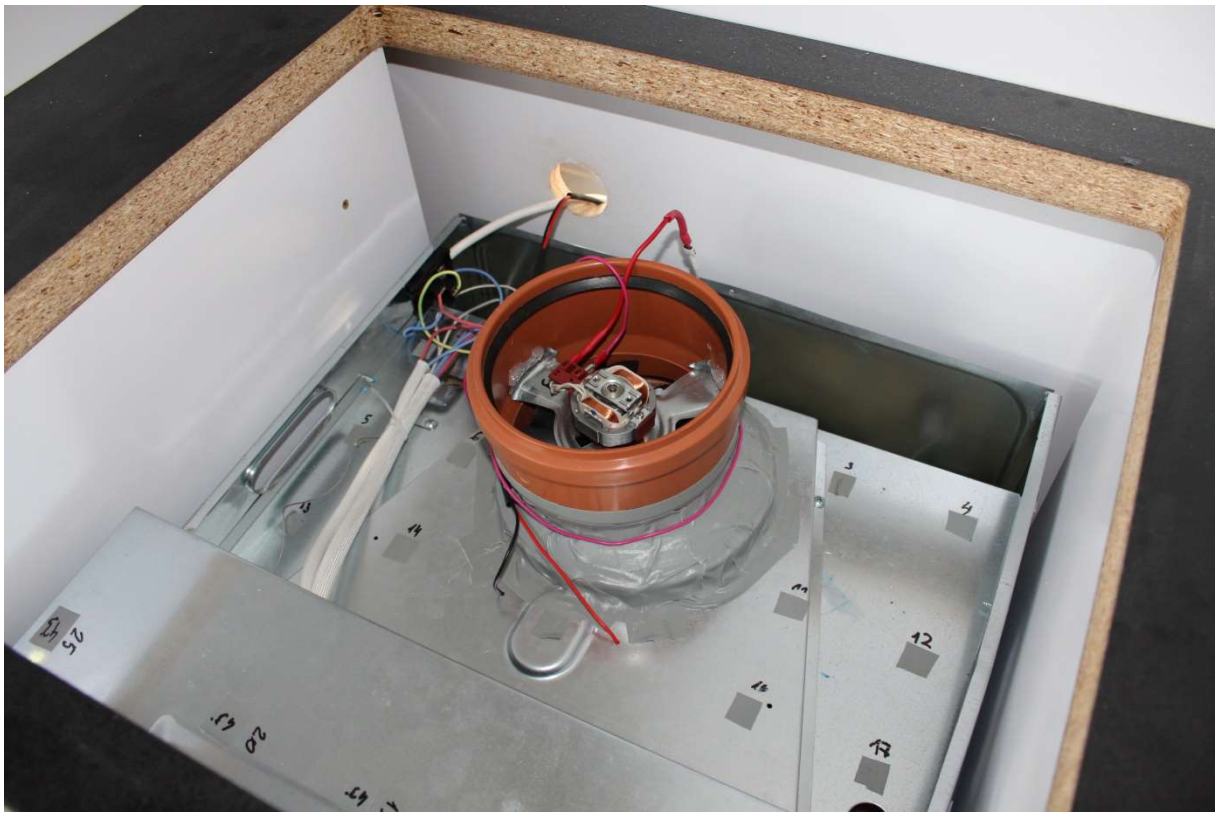
zdj 15. Działanie systemu - wciśnięcie płyty bezpieczeństwa.

Po dociśnięciu płyty zabezpieczającej do stelaża szafki wystaje ona nadal o ok. 1mm ponad korpus szafki - co przedstawiono na powyższym zdjęciu. Całość zamocowano do ściany za pomocą profili opracowanych przez Politechnikę Warszawską. Zamontowany zespół na ścianie przedstawia poniższe zdjęcie.

zdj 16. Zespół opuszczania i podnoszenia szafki zamontowany na ścianie.

Adaptacja systemu kogeneracji

W ramach prac dokonano adaptacji zakupionego sytemu kogeneracji. W demonstracyjnej zabudowie kuchennej zabudowano piekarnik wyposażony w ogniwa odzyskujące energię elektryczną z układu chłodzenia obudowy. Obecnie praktycznie każdy piekarnik wyposażony jest w system nadmuchu na rączkę zimnym powietrzem, które jednocześnie chłodzi górną część obudowy. Zdjęcie odkrytej obudowy piekarnika przedstawiono poniżej.

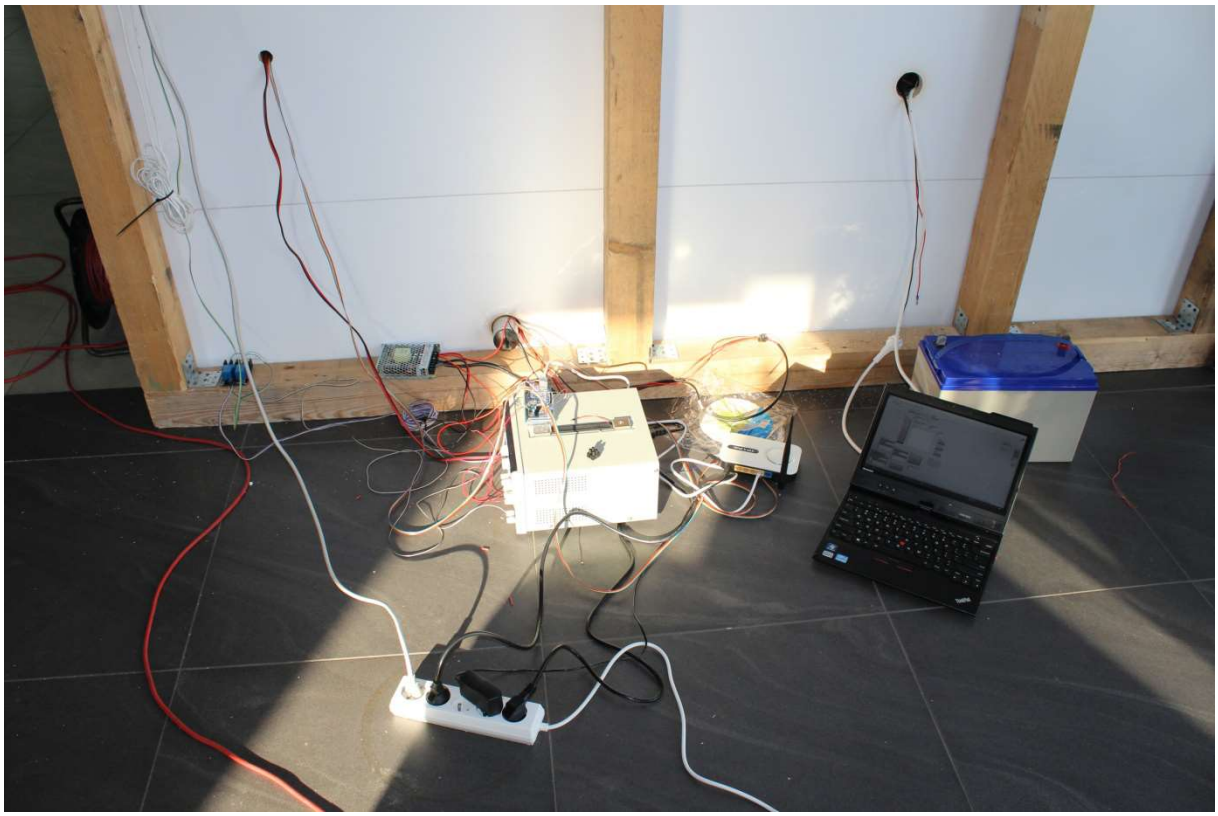


zdj 17. Górna część piekarnika z systemem odzyskiwania energii elektrycznej.

Dodatkowo w korpusie szafki wykonano specjalne kanały powietrzne doprowadzające ładunek świeżego powietrza z dolnej części zabudowy (kratka w dolnej osłonie przypodłogowej).

Kolejnym elementem systemu kogerenacyjnego było umieszczenie paneli fotowoltaicznych na dachu budynku.

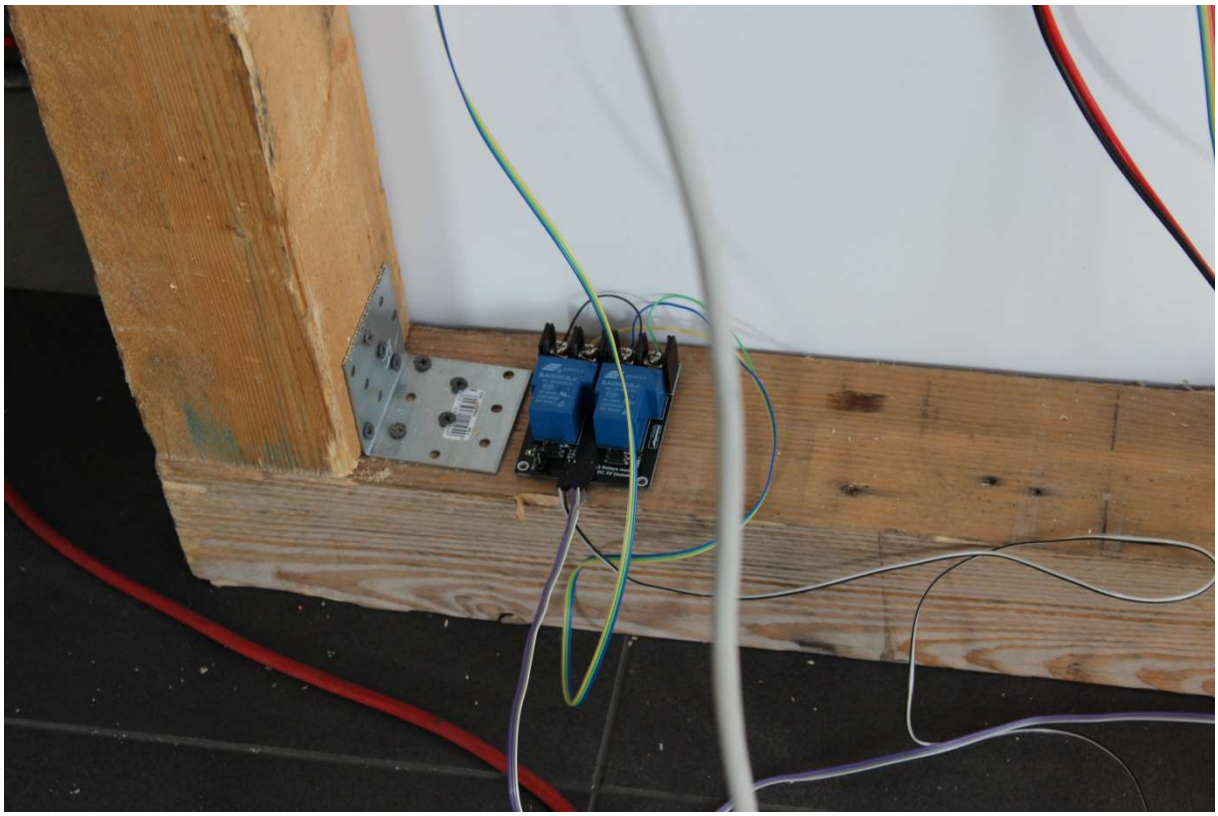
System został podłączony zgodnie z opracowaniem Politechniki Warszawskiej. Zasada działania opiera się na odbieraniu energii słonecznej i przekształcaniu jej w energię elektryczną. Energia elektryczna jest za pomocą przetwornika wykorzystywana do ładowania akumulatora, będącego integralną częścią systemu. Ogniwa termogeneratorów umieszczone w obudowie piekarnika, w przypadku gdy zaczynają pracować również przekazują energię elektryczną do magazynu energii w postaci akumulatora elektrochemicznego. Dopiero z akumulatora zasilane są elementy takie jak np. oświetlenie szafek.



zdj 18. System kogeneracji podczas podłączania

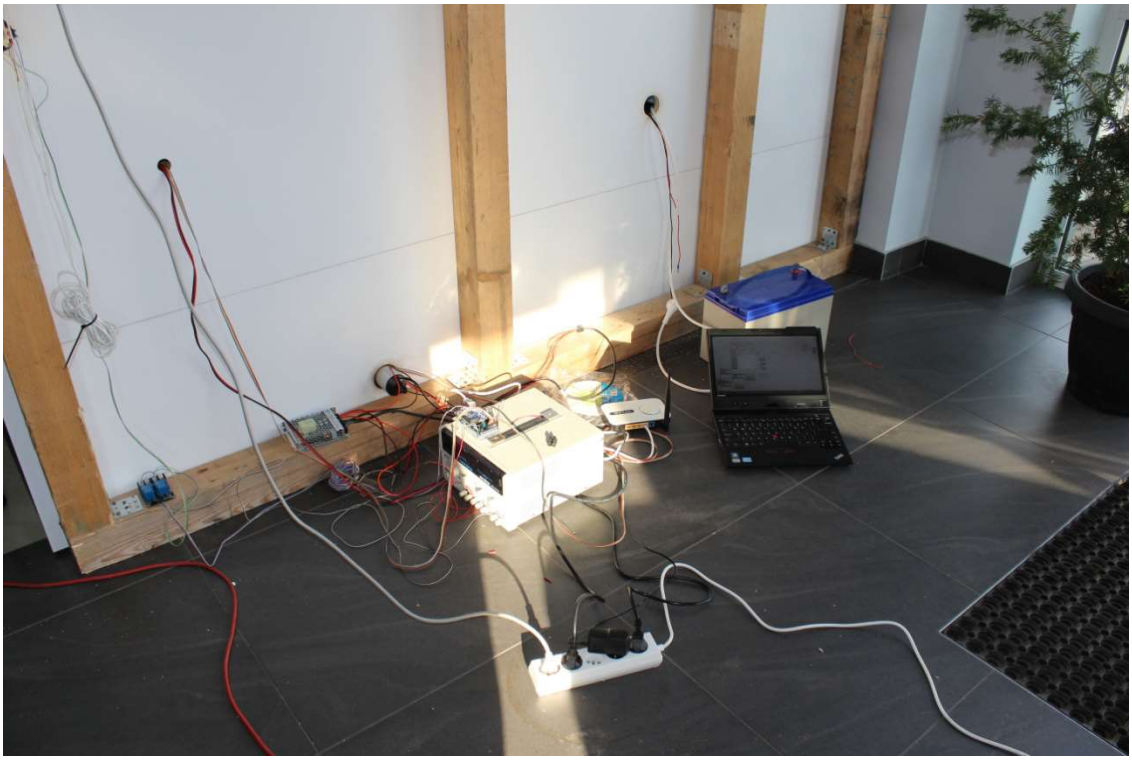
Adaptacja sterowania całością systemu

W ramach prac dokonano adaptacji układów sterujących całością systemu. Dokonano wymiany na większe przekaźników prądowych, tak aby zapewnić ich znaczące przewymiarowanie, a co za tym idzie maksymalnie wysoką bezawaryjność całości systemu. Przekaźniki podczas montażu przedstawiono na poniższym zdjęciu.



zdj 19. Przekaźniki podczas montażu

Dokonano również kalibracji całości systemu, co przedstawia poniższe zdjęcie.



zdj 20. Prace nad kalibracją systemu.

Oprócz powyższego dokonano również modyfikacji systemu łączenia WIFI, poprzez dołożenie dodatkowego routera nadającego własną sieć. Jest to rozwiązanie podyktowane chęcią ewentualnej rozbudowy w przyszłości systemu o zabudowy np. łazienkowe. W przypadku domu z kuchnią dostosowaną do naszego projektu, można zbudować drugi system działający w oddzielnej sieci do obsługi np. szafek łazienkowych. W ten sposób na jednym urządzeniu mobilnym, działając w dwóch sieciach (przełączanie sieci z poziomu aplikacji) możliwe będzie sterowanie dwoma oddzielnymi zabudowami. Poniższe zdjęcia przestawiają zabudowany system sterujący z dodatkowym routerem WIFI.

zdj 21. Całość sterowania systemem

Dodatkowo dokonano modyfikacji w postaci dołożenia dodatkowych, ręcznych klawiszy do opuszczania i podnoszenia szafki. Założono, ze nie musi pozostać możliwość ręcznego sterowania przynajmniej jednym modułem, a po drugie ma być to demonstrator który potencjalnemu Klientowi ma przybliżyć działanie systemu. Poniższe zdjęcia przedstawiają ścianę z zabudową kuchenną w pozycji otwartej i zamkniętej. Dodatkowo widoczny jest włącznik do ręcznego sterowania opuszczaniem i podnoszeniem szafki.

zdj 22. Szafki w pozycji złożonej, widoczny włącznik do sterowania.

zdj 23. Szafki w pozycji otwartej.

Adaptacja aplikacji sterującej

1. Zakres zmian w aplikacji wprowadzonych celem dostosowania do współpracy ze sterownikiem Arduino.

Aplikacja wymagała dostosowania do współpracy z wykorzystanym w projekcie sterownikiem Ardiuno.

Wprowadzono zmiany w zakresie:

- protokołu komunikacji,

- formatu komunikatów,

- częstotliwości wysyłania zapytań o status.

Wprowadzono również kilka poprawek, m. in.

- obsługę wyjątków w przypadku braku dostępnych sieci WiFi,

- odświeżanie ekranu - stan szafek

- obsługę oświetlenia.

W zakresie panelu użytkownika nie wprowadzono zmian.

W dalszej części opisano jedynie zmiany wprowadzone a kodzie źródłowym aplikacji podczas prac związanych z dostosowaniem jej do współpracy ze sterownikiem Arduino.

Nieopisane w tym zestawieniu elementy programu nie uległy zmianie.

2. Struktury danych

Aplikacja wykorzystuje zorganizowane struktury danych do przechowywania i przetwarzania danych związanych z układem szafek i oświetlenia oraz do wymiany informacji ze sterownikiem zewnętrznym. Struktury te to:

Plik konfiguracyjny

Plik konfiguracyjny jest tworzony przez zespół montujący system w kuchni i nie jest edytowalny przez użytkownika.

Składa się z dwóch modułów - nagłówka pliku oraz *n* elementów, po jednym dla każdego elementu systemu. Struktura nagłówka nie została zmieniona.

Dla każdego z elementów kuchni tworzy się jeden obiekt klasu `PskConfigFileItem`, zawierający szczegółowe dane o każdym z elementów systemu. W ramach dostosowania dodano pole: `itemPrevState`, które przechowuje dane o poprzednim stanie szafki. Jest ono wykorzystywane w przypadku wznawiania pracy po anulowaniu operacji otwierania lub zamykania, gdy element znajduje się w stanie idle.

```
public class PskConfigFileItem
{
    public int id;
    public string relayForward;
    public string relayBackward;
    public string edgeForward;
    public string edgeBackward;
    public int safetyPin;
    public string pinoutLight;
    public bool parentalLock;
    public string type { get; set; }
    public int xCoord;
    public int yCoord;
    public int width;
    public int height;
    public string itemState { get; set; }
    public string itemPrevState { get; set; } = "";
    public bool ledState { get; set; }
}
```

Dodatkowo zmieniono typ `itemState` na string. W bieżącej konfiguracji stan elementu jest przechowywany tekstowo, zgodnie z dokumentacją dostarczoną od projektanta oprogramowania zaimplementowanego na sterowniku Arduino.

Protokół transmisji danych

Ze względu na możliwości oraz złożoność programowania sterownika Arduino, zrezygnowano z przesyłania komunikatów w formacie JSON, na rzecz ciągu tekstowego par klucz - wartość.

Wprowadzono w kodzie aplikacji mechanizm konwertujący otrzymywane od sterownika dane na format JSON:

```
fileContent = "[" + fileContent + "]";

fileContent = fileContent.Replace("%&", "\",\"val\":\"");
fileContent = fileContent.Replace("&%", "\"},{\"key\":\"");
fileContent = fileContent.Replace("[%", "[{\"key\":\"");
fileContent = fileContent.Replace("&]", "\"}]");
```

W wyniku działania powyższej funkcji otrzymano dane w formacie JSON.

3. Komunikacja sieciowa

Aplikacja wykorzystuje do komunikacji ze sterownikiem systemu protokół UDP.

Komunikaty są przesyłane z wykorzystaniem zdefiniowanych portów (zmienionych w wyniku dostosowania aplikacji na: `61550` i `61551`).

Komunikacja inicjująca pracę systemu - po dostosowaniu do współpracy ze sterownikiem Arduino

Po uruchomieniu aplikacja wysyła komunikat `hello` sformatowany w JSON, którego celem jest wywołanie funkcji wysyłającej plik konfiguracyjny ze sterownika.

```
hello_0
```

W odpowiedzi aplikacja otrzymuje informację o liczbie części pliku konfiguracyjnego:

```
2
```

Następnie aplikacja wywołuje funkcję `hello`, celem otrzymania kolejnych fragmentów pliku konfiguracyjnego:

```
hello_1
```

```
hello_2
```

otrzymując w odpowiedzi *i-ty* fragment:

```
%liczbaSzaf%&4&%3_isLED%&TRUE&%3_type%&LED&%3_relayForward%&47&
%0_relayForward%&24&%0_relayBackward%&22&%0_edgeForward%&23&
%0_edgeBackward%&25&%0_safetyPin%&27&%0_type%&szuflada&%0_xCoord%&100&
%0_yCoord%&100&%1_relayForward%&28&%1_relayBackward%&30&%1_edgeForward
%&29&%1_edgeBackward%&31&%1_safetyPin%&33&%2_relayForward%&34&
%2_relayBackward%&36&%2_edgeForward%&35&%2_edgeBackward%&37&%2_safetyPin
%&39&
```

kontynuując komunikację aż do uzyskania `i = numOfItems` elementów pliku.

Następnie zawartość pliku jest łączona w jedną całość, formatowana do JSON i zamieniana na obiekty `PskConfigFileItem`.

Przesyłanie komunikatów wywołujących rozkazy akcji - po dostosowaniu do współpracy ze sterownikiem Arduino

Komunikaty wywołujące obsługę zdarzeń / akcji wysyłane są z wykorzystanie `id` - oznaczającym identyfikator szafki lub lampki która ma zostać uaktywniona oraz słowa kluczowego, np.:

```
2_forward
```

Cykliczne testowanie połączenia - po dostosowaniu do współpracy ze sterownikiem Arduino

Aplikacja cyklicznie sprawdza, czy sterownik zewnętrzny jest obecny w sieci, wysyłając komunikat `appPresent` . Jeśli przez 5 sekund nie otrzyma odpowiedzi, interpretuje ją jako zerwanie połączenia.

Komunikat App wygląda następująco:

```
status
```

Aplikacja powinna otrzymać w odpowiedzi ze sterownika w kolejnych pakietach UDP:

```
0_idle
```

```
1_backward
```

```
2_opened
```

```
3_idle
```

4. Kod źródłowy poszczególnych plików po wprowadzeniu zmian dostosowujących do współpracy ze sterownikiem Arduino

4.1 klasa `PskSettings` przechowująca główne ustawienia aplikacji oraz definiująca struktury danych

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace ProjektSzafkiKuchenne
{
    public class PskSettings
    {
        public string name = "nasza nazwa";
        //public int udpPortIn  = 11012;
        //public int udpPortOut = 11011;
        public int udpPortIn  = 61551;
        public int udpPortOut = 61550;
        public string udpAddress = "255.255.255.255";

        public bool configFileHeaderReceived = false;
        public bool configFileContentReceived = false;

        public int configFileTotalParts = 0;
        public int configFileStoreParts = 0;
        public int configFileLastQueryPartNo = 0;
        public string[] configFileParts { get; set; }

        public PskConfigFileHeader pskConfigFileHeader = new PskConfigFileHeader();
        public PskConfigFileItem[] tblItems { get; set; }


        public ImageSource iconConnectionOffline =
ImageSource.FromFile("ConnectionOffline_64x.png");
        public ImageSource iconConnecting =
ImageSource.FromFile("ConnectionWarning_64x.png");
        public ImageSource iconConnected =
ImageSource.FromFile("ConnectedGreen_64x.png");

        public enum myActions { forward, backward, cancel, on, off };

        public PskSettings()
        {



        }


        // ***********************************
        //
        // Klasy związane z plikiem konfiguracyjnym:
        //
        // ***********************************


        public class PskConfigFileHeader
```

```
        {
            public string wifiID;
            public string wifiPass;
            public int width;
            public int height;
            public int numOfItems;
        }


        // ************************************
        //
        // Klasy związane z obsługą szafek:
        //
        // ************************************

        public class PskConfigFileItem
        {
            public int id;
            public string relayForward;
            public string relayBackward;
            public string edgeForward;
            public string edgeBackward;
            public int safetyPin;
            public string pinoutLight;
            public bool parentalLock;
            public string type { get; set; }
            public int xCoord;
            public int yCoord;
            public int width;
            public int height;
            public string itemState { get; set; }
            public string itemPrevState { get; set; } = "";
            public bool ledState { get; set; }
        }



        // ************************************
        //
        // Klasy związane z obsługą transmisji:
        //
        //  wysyłane do sterownika szafek
        //  key:
        //   hello - wysyła żądanie otrzymania pliku konfiguracyjnego
        //
        // ************************************


        public struct PskEvents
        {
            public int id;          // identyfikator szafki lub lampki
            public string key;      // forward, backward, cancel, status
            public string value;
        }



    }
}
```

4.2. Klasa `Communication` wykorzystywana do prowadzenia komunikacji sieciowej UDP pomiędzy aplikacją a sterownikiem szafek:

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace ProjektSzafkiKuchenne
{
    public class Communication
    {

        // ***************************************************
        //
        // wysłanie żądania otrzymania pliku konfiguracyjnego
        //
        // ***************************************************

        public bool otrzymanoOdpowiedz;

        public Communication()
        {


        }


        public void myTimerStart()
        {
            Device.StartTimer(new TimeSpan(0, 0, 0, 2, 500), () =>
            {
                if (App.myApp.myAppOnTop)
                {

                    if (App.myApp.settings.configFileHeaderReceived == false)
                    {
                        Task.Run(() => {
                            SendHello();
                        });
                    }
                    else if ((App.myApp.settings.configFileHeaderReceived == true) &&
(App.myApp.settings.configFileContentReceived == false))
                    {
                        DateTime dtNow = DateTime.Now;
                        var diffInSeconds = (dtNow -
App.myApp.connection.lastTimeReceivedData).TotalMilliseconds;
                        if (diffInSeconds > 2000)
                        {
                            Task.Run(() => {
                                CheckConfigFileItems();
                            });
                        }
                    }
                    else
                    {
                        Task.Run(() => {
                            // cykliczne sprawdzanie, czy nie utracono połączenia
```

```
                            appPresent();
                            DateTime dtNow = DateTime.Now;
                            var diffInSeconds = (dtNow -
App.myApp.connection.lastTimeReceivedData).TotalMilliseconds;
                            if (diffInSeconds < 5000)
                            {
                                App.myApp.connection.StateCrt =
Connection.ConnectionState.Connected;
                            }
                            else
                            {
                                App.myApp.connection.StateCrt =
Connection.ConnectionState.Disconnected;
                            }
                        });
                    }

                    return true;
                }
                return false;
            }
            );
        }


        public void SendHello(int i = 0)
        {
            // i - numer części pliku, 0 = nagłówek, 1..n - numer części pliku

            // AppCarDataBaseContent tmpDBContent =
JsonConvert.DeserializeObject<AppCarDataBaseContent>(fcontent);
            // fcontent = JsonConvert.SerializeObject(tmpDBContent);

            App.myApp.settings.configFileContentReceived = false;
            App.myApp.connection.StateCrt = Connection.ConnectionState.Connecting;

            //PskSettings.PskEvents myEvent = new PskSettings.PskEvents() { id = i,
key = "hello", value = null };
            //string msg = JsonConvert.SerializeObject(myEvent);
            //App.myApp.connection.udpSendMessage(msg);

            App.myApp.connection.udpSendMessage("hello_" + i.ToString());

            Debug.WriteLine( "hello_" + i.ToString() );

        }

        //public void ReceivedHello(int id, string key, string value)
        public void ReceivedHello(string data)
        {
            // otrzymano nagłówek pliku konfiguracyjnego
            if (App.myApp.settings.configFileLastQueryPartNo > 0)
            {
                try
                {
                    // zapisanie pliku konfiguracyjnego
                    App.myApp.settings.configFileHeaderReceived = true;
                }
                catch { }
            }
```

```
            CheckConfigFileItems();
        }

        // sprawdzenie kompletności pliku konfiguracyjnego i wysłanie zapytania o
pierwszy brakujący fragment, jeśli taki występuje
        public void CheckConfigFileItems()
        {
            bool configFileCompleted = true;
            try
            {

                if (App.myApp.settings.configFileStoreParts <
App.myApp.settings.configFileTotalParts)
                {
                    configFileCompleted = false;
                    App.myApp.settings.configFileLastQueryPartNo =
App.myApp.settings.configFileStoreParts + 1;
                    SendHello( App.myApp.settings.configFileLastQueryPartNo );
                }

                // plik konfiguracyjny jest kompletny
                if (configFileCompleted && !
App.myApp.settings.configFileContentReceived)
                {
                    string abc = App.myApp.settings.configFileParts[0] +
App.myApp.settings.configFileParts[1];

                    App.myApp.settings.configFileContentReceived = true;
                    App.myApp.connection.StateCrt =
Connection.ConnectionState.Connected;

                    if (MergeConfigFile())
                    {

                        Device.BeginInvokeOnMainThread(() =>
                        {
                            App.myApp.myMainPage.renderMyView();
                        });

                        App.myApp.connection.StateCrt =
Connection.ConnectionState.Connected;

                    }

                }

            }
            catch { }
        }


        public bool MergeConfigFile()
        {
            bool retval = false;

            try
            {
                Debug.WriteLine("plik konfiguracyjny jest kompletny, rozpoczynam jego
analizę! ");

                string fileContent = String.Empty;
                foreach (string a in App.myApp.settings.configFileParts)
```

```
                {
                    fileContent += a;
                }

                fileContent = "[" + fileContent + "]";

                fileContent = fileContent.Replace("%&", "\",\"val\":\"");
                fileContent = fileContent.Replace("&%", "\"},{\"key\":\"");
                fileContent = fileContent.Replace("[%", "[{\"key\":\"");
                fileContent = fileContent.Replace("&]", "\"}]");

                configFileStruct[] tblOfConfig =
JsonConvert.DeserializeObject<configFileStruct[]>(fileContent);

                foreach (configFileStruct item in tblOfConfig)
                {

                    switch (item.key)
                    {
                        case "ln":
                            // liczba elementów graficznych na ekranie
                            int number0;
                            bool result0 = Int32.TryParse(item.val, out number0);
                            if (result0)
                            {
                                App.myApp.settings.pskConfigFileHeader.numOfItems =
number0;

                                App.myApp.settings.tblItems = new
PskSettings.PskConfigFileItem[number0];
                                for (int j=0; j < number0; j++)
                                {
                                    App.myApp.settings.tblItems[j] = new
PskSettings.PskConfigFileItem();
                                    App.myApp.settings.tblItems[j].id = j;
                                    try
                                    {

App.myApp.myMainPage.myMainPageViewModel.stanSzafek.Add(new
ViewModels.MainPageViewModel.MojeSzafki() { Value = "closed", type = "" });
                                    }
                                    catch
                                    {
                                    }
                                }
                            }
                            break;
                        case "liczbaSzaf":
                            // liczba elementów aktywnych
                            break;
                        case "width":
                            App.myApp.settings.pskConfigFileHeader.width =
Int32.Parse(item.val);
                            break;
                        case "height":
                            App.myApp.settings.pskConfigFileHeader.height =
Int32.Parse(item.val);
                            break;
                        default:

                            if (item.key.Length > 2)
                            {
```

```
                                if (item.key.Substring(1,1).CompareTo("_") == 0)
                                {
                                    // mamy tutaj parametr szafki lub lampki aktywnej

                                    int number1;

                                    bool result1 =
Int32.TryParse(item.key.Substring(0, 1), out number1);
                                    if (result1)
                                    {

                                        if (App.myApp.settings.tblItems.Length >
number1)
                                        {

                                            switch (item.key.Substring(2))
                                            {
                                                case "relayForward":

App.myApp.settings.tblItems[number1].relayForward = item.val;
                                                    break;
                                                case "relayBackward":

App.myApp.settings.tblItems[number1].relayBackward = item.val;
                                                    break;
                                                case "edgeForward":

App.myApp.settings.tblItems[number1].edgeForward = item.val;
                                                    break;
                                                case "edgeBackward":

App.myApp.settings.tblItems[number1].edgeBackward = item.val;
                                                    break;
                                                case "safetyPin":

App.myApp.settings.tblItems[number1].safetyPin = Int32.Parse(item.val);
                                                    break;
                                                case "type":

App.myApp.settings.tblItems[number1].type = item.val;

App.myApp.myMainPage.myMainPageViewModel.stanSzafek[number1].type = item.val;
                                                    break;
                                                case "xCoord":

App.myApp.settings.tblItems[number1].xCoord = Int32.Parse(item.val);
                                                    break;
                                                case "yCoord":

App.myApp.settings.tblItems[number1].yCoord = Int32.Parse(item.val);
                                                    break;
                                                case "width":

App.myApp.settings.tblItems[number1].width = Int32.Parse(item.val);
                                                    break;
                                                case "height":

App.myApp.settings.tblItems[number1].height = Int32.Parse(item.val);
                                                    break;
                                                case "isLED":

//App.myApp.settings.tblItems[number1]. = Int32.Parse(item.val);
```

```
                                        break;
                                    default:
                                        break;
                                }

                            }

                        }
                        else
                        {

                            //błąd

                        }

                    }
                    else if (item.key.Substring(2, 1).CompareTo("_") == 0)
                    {
                        // mamy tutaj parametr elementu nieaktywnego

                        int number1;

                        bool result1 =
Int32.TryParse(item.key.Substring(0, 2), out number1);
                        if (result1)
                        {

                            if (App.myApp.settings.tblItems.Length >
number1)
                            {

                                switch (item.key.Substring(3))
                                {
                                    case "relayForward":

App.myApp.settings.tblItems[number1].relayForward = item.val;
                                        break;
                                    case "relayBackward":

App.myApp.settings.tblItems[number1].relayBackward = item.val;
                                        break;
                                    case "edgeForward":

App.myApp.settings.tblItems[number1].edgeForward = item.val;
                                        break;
                                    case "edgeBackward":

App.myApp.settings.tblItems[number1].edgeBackward = item.val;
                                        break;
                                    case "safetyPin":

App.myApp.settings.tblItems[number1].safetyPin = Int32.Parse(item.val);
                                        break;
                                    case "type":

App.myApp.settings.tblItems[number1].type = item.val;
                                        break;
                                    case "xCoord":

App.myApp.settings.tblItems[number1].xCoord = Int32.Parse(item.val);
                                        break;
                                    case "yCoord":
```

```
App.myApp.settings.tblItems[number1].yCoord = Int32.Parse(item.val);
                                                    break;
                                                case "width":

App.myApp.settings.tblItems[number1].width = Int32.Parse(item.val);
                                                    break;
                                                case "height":

App.myApp.settings.tblItems[number1].height = Int32.Parse(item.val);
                                                    break;
                                                case "isLED":

//App.myApp.settings.tblItems[number1]. = Int32.Parse(item.val);
                                                    break;
                                                default:
                                                    break;
                                            }

                                        }

                                    }
                                    else
                                    {

                                        //błąd

                                    }

                                }
                            }

                            break;
                        }

                    }

                    retval = true;
                }
                catch { }

                return retval;

            }

            public struct configFileStruct
            {
                public string key;
                public string val;
            }



            public void appPresent()
            {
                // wysłanie komunikatu potwierdzającego aktywność aplikacji oraz
wywołującego odowiedź systemu
                //PskSettings.PskEvents myEvent = new PskSettings.PskEvents() { id = 0,
key = "appPresent", value = null };
                //string msg = JsonConvert.SerializeObject(myEvent);

                App.myApp.connection.udpSendMessage("status");
```

```
        }


        //public void sendAction(int id, PskSettings.myActions action)
        public void sendAction(string action)
        {

            App.myApp.connection.udpSendMessage(action);

        }




    }
}
```

4.3. Klasa `Connection` do obsługi protokołu UDP

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Newtonsoft.Json;
using Sockets.Plugin;
using System.Diagnostics;

namespace ProjektSzafkiKuchenne
{
    public class Connection
    {
        public enum ConnectionState { Disconnected, Connecting, Connected };

        public ConnectionState stateTgt = ConnectionState.Disconnected;   // żądany
stan połączenia ze sterownikiem

        public ConnectionState stateCrt { get; set; } = ConnectionState.Disconnected;
// bieżący stan połączenia ze sterownikiem
        public ConnectionState StateCrt {
            set
            {
                switch (value)
                {
                    case ConnectionState.Connected:
                        App.myApp.myMainPage.myMainPageViewModel.IconFile =
App.myApp.settings.iconConnected;
                        break;
                    case ConnectionState.Connecting:
                        App.myApp.myMainPage.myMainPageViewModel.IconFile =
App.myApp.settings.iconConnecting;
                        break;
                    case ConnectionState.Disconnected:
                        App.myApp.myMainPage.myMainPageViewModel.IconFile =
App.myApp.settings.iconConnectionOffline;
                        break;
                    default:
                        App.myApp.myMainPage.myMainPageViewModel.IconFile =
App.myApp.settings.iconConnectionOffline;
                        break;
                }
                stateCrt = value;
            }
            get { return stateCrt; }
        }

        public DateTime lastTimeReceivedData = DateTime.Now;

        UdpSocketReceiver receiver = new UdpSocketReceiver();
        UdpSocketClient client = new UdpSocketClient();

        public Connection()
        {

            receiver.MessageReceived += (sender, args) =>
                {
                    // handler - odebrano dane przez udp
```

```
                    string from = String.Format("{0}:{1}", args.RemoteAddress,
args.RemotePort);
                    var data = Encoding.UTF8.GetString( args.ByteData, 0,
args.ByteData.Length );
                    Debug.WriteLine(from + " " + data);

                    // zapisanie znacznika czasu uzyskania najnowszej odpowiedzi
                    lastTimeReceivedData = DateTime.Now;

                    try
                    {
                        //PskSettings.PskEvents formatedData =
JsonConvert.DeserializeObject<PskSettings.PskEvents>(data);
                        //formatedData.key

                        // oczekiwanie na pierwszą część - nagłówek pliku
konfiguracyjnego
                        if (!App.myApp.settings.configFileHeaderReceived)
                        {
                            if (data.Length == 1)
                            {
                                int number;

                                bool result = Int32.TryParse(data, out number);
                                if (result)
                                {

                                    Debug.WriteLine("Odebrano naglowek:" + data + " "
+ number.ToString());

                                    App.myApp.settings.configFileTotalParts = number;
                                    App.myApp.settings.configFileHeaderReceived =
true;

                                    App.myApp.settings.configFileParts = new
string[number];

                                    App.myApp.communication.CheckConfigFileItems();
                                }
                                else
                                {
                                    Debug.WriteLine("Attempted conversion of '{0}'
failed.", data == null ? "<null>" : data);
                                }

                            }
                        }
                        else if (App.myApp.settings.configFileHeaderReceived && !
App.myApp.settings.configFileContentReceived &&
(App.myApp.settings.configFileLastQueryPartNo > 0))
                        {

                            Debug.WriteLine("Odebrano czesc: " +
App.myApp.settings.configFileLastQueryPartNo.ToString() + " " + data);
                            // prawdopodobnie odebrano fragment pliku konfiguracyjnego

App.myApp.settings.configFileParts[ App.myApp.settings.configFileLastQueryPartNo - 1 ]
= data;

                            App.myApp.settings.configFileStoreParts =
App.myApp.settings.configFileLastQueryPartNo;
```

```
                            App.myApp.communication.CheckConfigFileItems();

                        }
                        else
                        {
                            // uaktualnienie statusu:
                            if (data.Length > 2)
                            {
                                int number;

                                bool result = Int32.TryParse( data.Substring(0, 1) ,
out number);
                                if (result)
                                {
                                    if
((App.myApp.settings.tblItems[number].itemPrevState !=
App.myApp.settings.tblItems[number].itemState) &&
(App.myApp.settings.tblItems[number].itemState != null))
                                    {
                                        if
((App.myApp.settings.tblItems[number].itemState.CompareTo("forward") == 0) ||
(App.myApp.settings.tblItems[number].itemState.CompareTo("backward") == 0))

App.myApp.settings.tblItems[number].itemPrevState =
App.myApp.settings.tblItems[number].itemState;
                                    }
                                    App.myApp.settings.tblItems[number].itemState =
data.Substring(2);

App.myApp.myMainPage.myMainPageViewModel.stanSzafek[number].Value = data.Substring(2);
                                }

                            }

                        }

                    }
                    catch { }

                };

        }

        public async void udpSendMessage(string msg)
        {
            // wyślij do address:port
            var msgBytes = Encoding.UTF8.GetBytes(msg);
            try
            {
                await client.SendToAsync(msgBytes, App.myApp.settings.udpAddress,
App.myApp.settings.udpPortOut);
            }
            catch
            {
                Debug.WriteLine("brak sieci ?");
            }
        }

        public async void udpStartListening()
        {
            // rozpocznij nasłuchiwanie na zadanym procie
            await receiver.StartListeningAsync( App.myApp.settings.udpPortIn );
```

```
            }

        }
    }
```

4.4. Klasa App (bez zmian)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Xamarin.Forms;

namespace ProjektSzafkiKuchenne
{
    public partial class App : Application
    {
        public PskSettings settings = new PskSettings();
        public Connection connection = new Connection();
        public Communication communication = new Communication();
        public MainPage myMainPage;
        public static App myApp;
        public bool myAppOnTop;

        public App()
        {
            InitializeComponent();
            myApp = this;
            myMainPage = new MainPage();
            MainPage =  new NavigationPage( myMainPage );
        }

        protected override void OnStart()
        {
            // Wywoływane kiedy aplikacja jest uruchamiana
            myAppOnTop = true;
            communication.myTimerStart();
        }

        protected override void OnSleep()
        {
            // Wywoływane kiedy aplikacja jest usypiana
            myAppOnTop = false;
        }

        protected override void OnResume()
        {
            // Wywoływane kiedy aplikacja jest wznawiana
            myAppOnTop = true;
            communication.myTimerStart();
        }
    }
}
```

4.5. Klasa `MainPage` będąca Modelem dla głównego ekranu aplikacji

```
using System;
using System.Net;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using Sockets.Plugin;
using Newtonsoft.Json;
using System.Diagnostics;
using System.Globalization;

namespace ProjektSzafkiKuchenne
{
    public partial class MainPage : ContentPage
    {

        public ViewModels.MainPageViewModel myMainPageViewModel;

        public MainPage()
        {
            myMainPageViewModel = new ViewModels.MainPageViewModel();
            this.BindingContext = myMainPageViewModel;

            InitializeComponent();

            App.myApp.connection.udpStartListening();

            //Task.Run(() => {
            //    renderMyView()
            //});

        }


        private void Button_Clicked_1(object sender, EventArgs e)
        {
            App.myApp.communication.SendHello();
        }


        // ------------------------------------------------------
        // wyświetlenie widoku szafek
        // ------------------------------------------------------

        public void renderMyView()
        {
            try
            {

                var colDef = new ColumnDefinitionCollection();
                for (int i = 0; i <= App.myApp.settings.pskConfigFileHeader.width ; i+
+) colDef.Add(new ColumnDefinition { Width = new GridLength(1.6 ,
GridUnitType.Absolute) });

                var rowDef = new RowDefinitionCollection();
```

```
                for (int i = 0; i <= App.myApp.settings.pskConfigFileHeader.height ;
i++) rowDef.Add(new RowDefinition { Height = new GridLength(1.6 ,
GridUnitType.Absolute) });

                Grid grid = new Grid
                {
                    VerticalOptions = LayoutOptions.FillAndExpand,
                    HorizontalOptions = LayoutOptions.FillAndExpand,
                    ColumnSpacing = 0,
                    RowSpacing = 0,
                    RowDefinitions = rowDef,
                    ColumnDefinitions = colDef,
                };

                foreach (PskSettings.PskConfigFileItem item in
App.myApp.settings.tblItems)
                {

                    var btn = new Button
                    {
                        Text = "", // item.id.ToString(),
                        FontSize = Device.GetNamedSize(NamedSize.Large,
typeof(Label)),
                        HorizontalOptions = LayoutOptions.FillAndExpand,
                        CommandParameter = item.id
                    };

                    //btn.BindingContext = App.myApp.settings; //[ item.id ];

                    btn.BindingContext = myMainPageViewModel;
                    btn.SetBinding(Button.BackgroundColorProperty, "stanSzafek["+
item.id + "].Value", BindingMode.TwoWay, new
TypeToNameConverter( myMainPageViewModel.stanSzafek[ item.id ] ) );

                    //if (App.myApp.settings.tblItems[item.id].type != null)
                    //btn.SetBinding(Button.BackgroundColorProperty, "tblItems",
BindingMode.TwoWay, new
TypeToNameConverter( App.myApp.settings.tblItems[ item.id ] ) );

                    btn.Clicked += btnItem_Clicked;

                    if (item.height > 0 && item.width > 0)
                        grid.Children.Add(btn, item.xCoord, item.xCoord+item.width,
item.yCoord, item.yCoord+item.height);

                }

                Debug.WriteLine("s");

                myScrollView.Content = grid;

            }
            catch
            {

            }

        }


        public class TypeToNameConverter : IValueConverter
        {
```

```
            ViewModels.MainPageViewModel.MojeSzafki item;

            public TypeToNameConverter(ViewModels.MainPageViewModel.MojeSzafki item)
            {
                this.item = item;
            }

            public object Convert(object value, Type targetType, object parameter,
CultureInfo culture)
            {
                if (item.type.CompareTo("LED") == 0)
                {

                    switch (item.Value)
                    {
                        case "forward":
                            return Color.Yellow;
                            break;
                        case "backward":
                            return Color.Orange;
                            break;
                        default:
                            return Color.Orange;
                            break;
                    }

                }
                else
                {
                    switch (item.Value)
                    {
                        case "idle":
                            return Color.DarkGray;
                            break;
                        case "emergency":
                            return Color.RosyBrown;
                            break;
                        case "forward":
                            return Color.DarkGreen;
                            break;
                        case "backward":
                            return Color.SandyBrown;
                            break;
                        case "opened":
                            return Color.Green;
                            break;
                        case "closed":
                            return Color.Gray;
                            break;
                        default:
                            return Color.Gray;
                            break;
                    }

                }
            }

            public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
            {
                return Color.Beige; // (value / GetParameter(parameter));
            }
```

```
            double GetParameter(object parameter)
            {
                if (parameter is double)
                    return (double)parameter;

                else if (parameter is int)
                    return (int)parameter;

                else if (parameter is string)
                    return double.Parse((string)parameter);

                return 1;
            }
        }


        // ---------------------------------------------------------------
        // wywołanie zdarzenia po kliknięciu elementu: szafki lub oświetlenia
        // ---------------------------------------------------------------

        private void btnItem_Clicked(object sender, EventArgs e)
        {
            var button = (Button)sender;
            var idv = button.CommandParameter;
            byte id = Convert.ToByte(idv);


            PskSettings.PskConfigFileItem item = App.myApp.settings.tblItems.First(x
=> x.id == id);

            if (item.type != null)
            {
                // sprawdzenie, czy konieczne jest podanie PIN
                if (item.parentalLock) {
                    OnPinRequest(item);
                }
                else
                {
                    actionConfirmed(item);
                }
            }
        }

        public void actionConfirmed(PskSettings.PskConfigFileItem item)
        {
            string targetAction;

            // ustalenie akcji
            if ((item.type.CompareTo("szafka") == 0) ||
(item.type.CompareTo("szuflada") == 0) || (item.type.CompareTo("podnosnik") == 0))
            {

                switch (item.itemState)
                {
                    case "idle": //
                        if (item.itemPrevState.CompareTo("forward") == 0)
                            targetAction = "backward";
                        else
                            targetAction = "forward";
                        break;
                    case "emergency": //
```

```
                        targetAction = "cancel";
                        break;
                    case "forward": //
                        targetAction = "cancel";
                        break;
                    case "backward": //
                        targetAction = "cancel"; //2
                        break;
                    case "opened": //
                        targetAction = "backward";
                        break;
                    case "closed": //
                        targetAction = "forward";
                        break;
                    default: // stan rezerwowy
                        targetAction = "cancel";
                        break;
                }

            }
            else if (item.type.CompareTo("LED") == 0)
            {
                switch (item.itemState)
                {
                    case "forward": // jest ON, zmieniamy na off
                        targetAction = "backward";
                        break;
                    case "backward": // jest OFF, zmieniamy na on
                        targetAction = "forward";
                        break;
                    default: // stan rezerwowy
                        targetAction = "backward";
                        break;
                }
            }
            else
            {
                // stan rezerwowy
                targetAction = "cancel";
            }

            // zmieniamy stan w pamięci aplikacji, zakładamy że sterownik odbiera
komunikat
            if (App.myApp.settings.tblItems[item.id].itemPrevState !=
App.myApp.settings.tblItems[item.id].itemState)
            {
            //    App.myApp.settings.tblItems[item.id].itemPrevState =
App.myApp.settings.tblItems[item.id].itemState;
            }

            //App.myApp.settings.tblItems[item.id].itemState = targetAction;
            App.myApp.myMainPage.myMainPageViewModel.stanSzafek[item.id].Value =
targetAction;

            Task.Run(() => {
                App.myApp.communication.sendAction( item.id.ToString() + "_" +
targetAction );
            });

        }
```

```
        // ------------------------------------------------------
        // wywołanie strony O programie
        // ------------------------------------------------------

        private void btnToolBarAboutApp_Clicked(object sender, EventArgs e)
        {
            Navigation.PushAsync(new Views.AboutPage());
        }

        async void OnPinRequest(PskSettings.PskConfigFileItem item)
        {
          await Navigation.PushModalAsync ( new Views.pinDialog(item) );
        }


    }
}
```

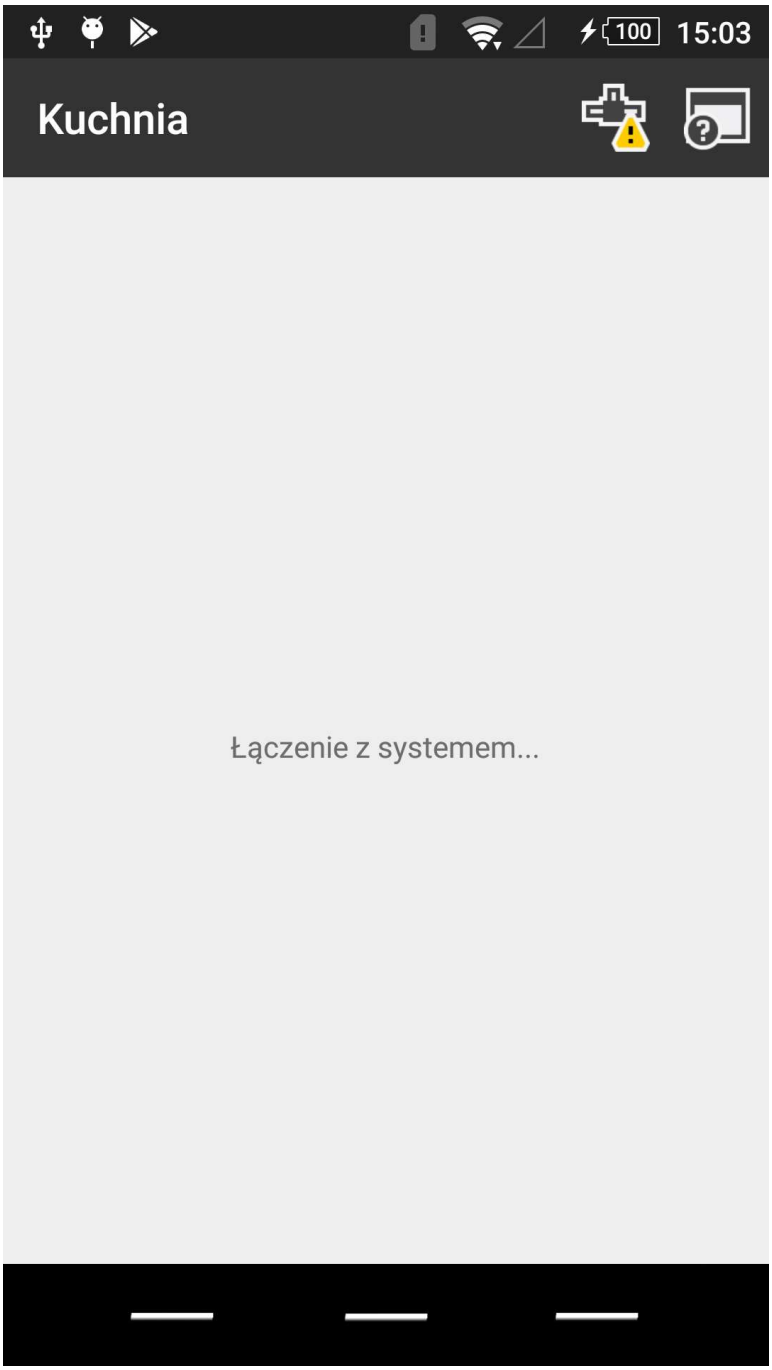
4.6. Kod XAML okien aplikacji

Kod XAML żadnego z okien nie uległ modyfikacji.

Instrukcja obsługi aplikacji sterującej

Instrukcja obsługi aplikacji

Po uruchomieniu aplikacji otwiera się główne okno programu.



Rysunek 1: Główne okno programu podczas nawiązywania połączenia ze sterownikiem

W tym czasie w tle aplikacja nawiązuje połączenie ze sterownikiem i pobiera od niego plik konfiguracyjny.

W przypadku braku sieci Wi-Fi aplikacja oczekuje na połączenie z siecią.

W górnej części okna po prawej stronie na pasku stanu znajdują się dwie ikony.

Rysunek 2: Pasek stanu w górnej części aplikacji

Pierwsza z nich symbolizuje stan połączenia ze sterownikiem systemu.

Może ona przyjąć trzy stany:

Oznacza proces łączenia się ze sterownikiem w celu pobrania pliku konfiguracyjnego.

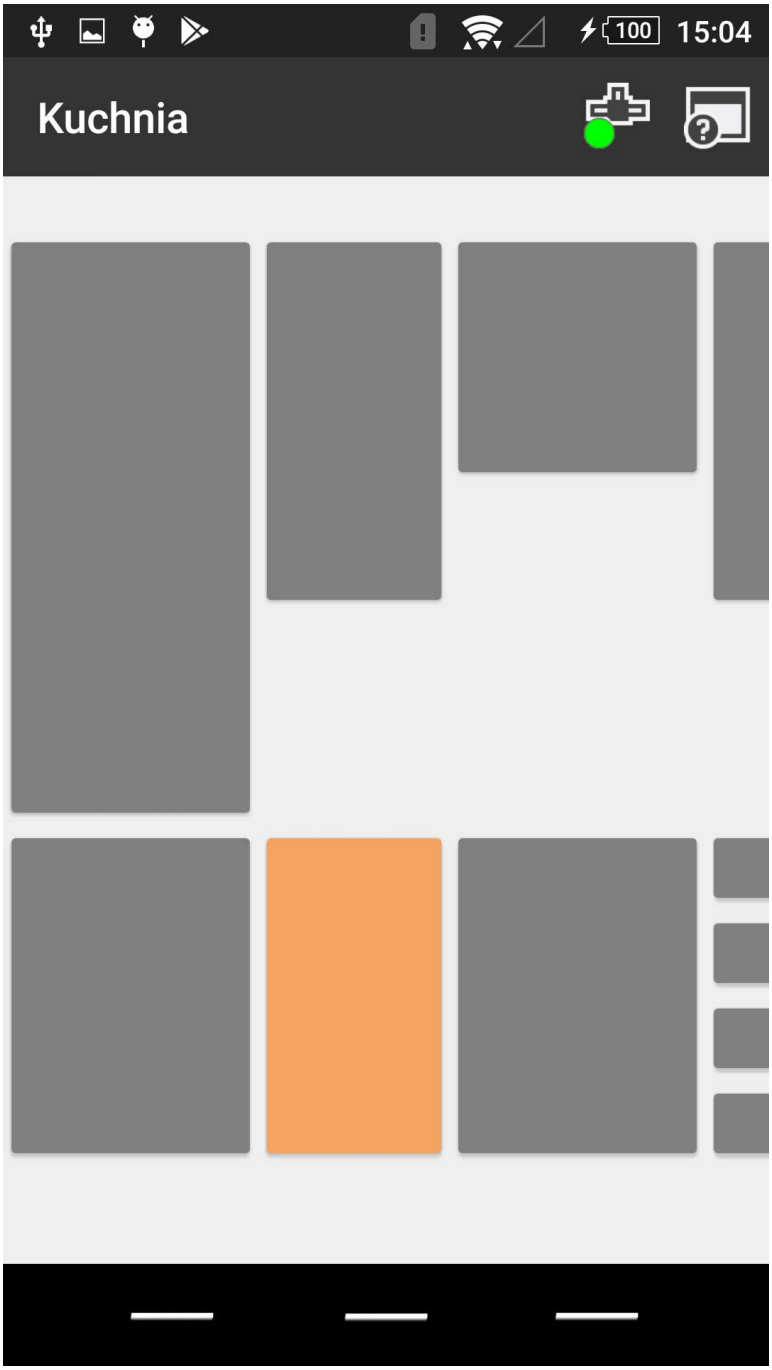
Oznacza normalną pracę systemu - aktywne połączenie

Oznacza brak komunikacji ze sterownikiem systemu - możliwe przyczyny to brak komunikacji sieciowej lub wyłączony sterownik.

Druga ikona wywołuje okienko z pomocą i opisem programu.

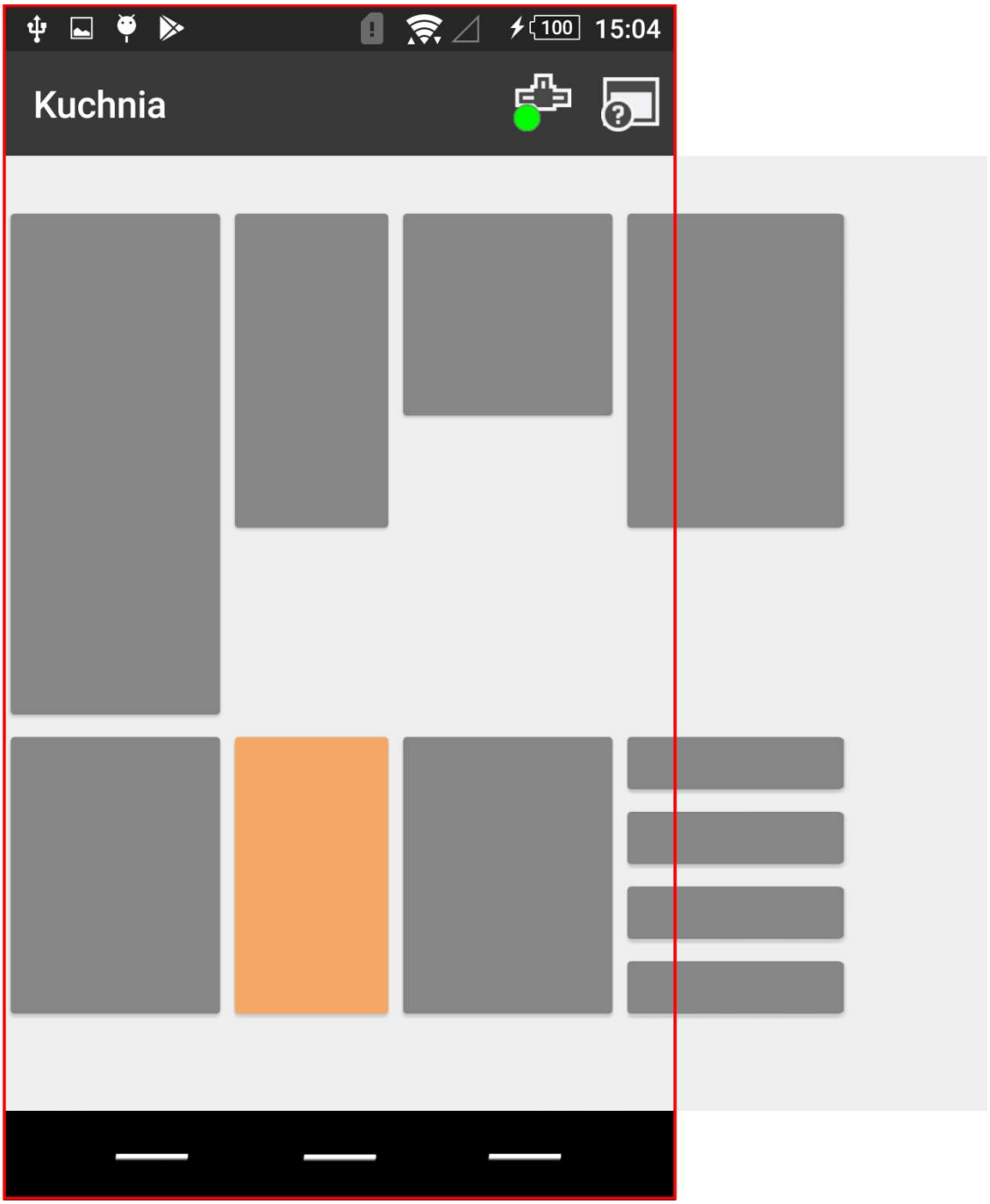
Otwarcie okienka z pomocą i opisem programu

Po poprawnym załadowaniu pliku konfiguracyjnego aplikacja wyświetla okno z wizualizacją kuchni.



Rysunek 3: Główne okno aplikacji

Ze względu na pionową orientację ekranu, wizualizacja kuchni znajduje się w horyzontalnie przewijalnym boksie.



Rysunek 4: Obszar widoczny na ekranie oraz cała wizualizacja kuchni, dostępna za pomocą przewijanego widoku

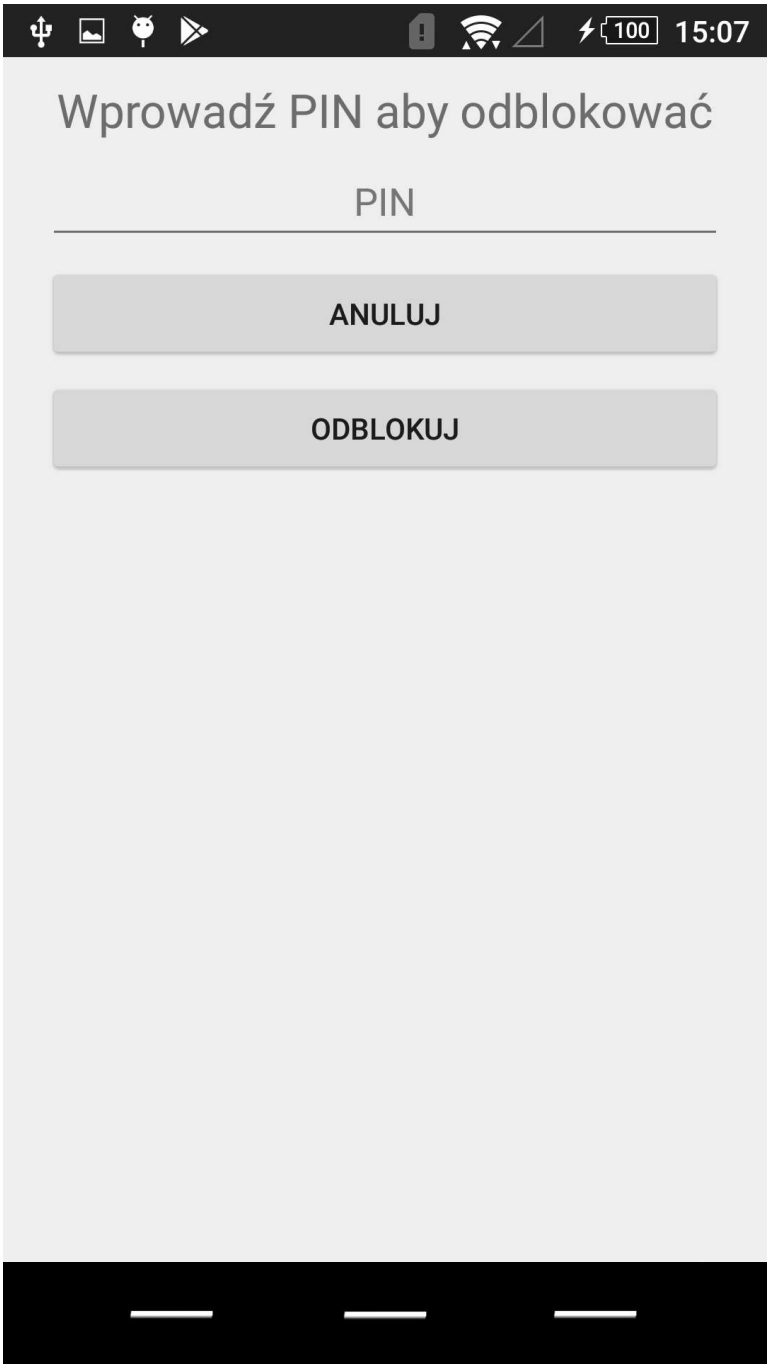
Kolor szafek może być konfigurowalny z poziomu pliku konfiguracyjnego.

Szafka, która jest otwarta jest wyróżniona innym kolorem.

Kolor szary - szafka zamknięta lub nieaktywna,
Kolor ciemnozielony - szafka otwiera się,
Kolor pomarańczowy - szafka zamyka się,
Kolor zielony - szafka otwarta.

Aby otworzyć lub zamknąć szafkę wystarczy kliknąć w jej rysunek na wizualizacji.

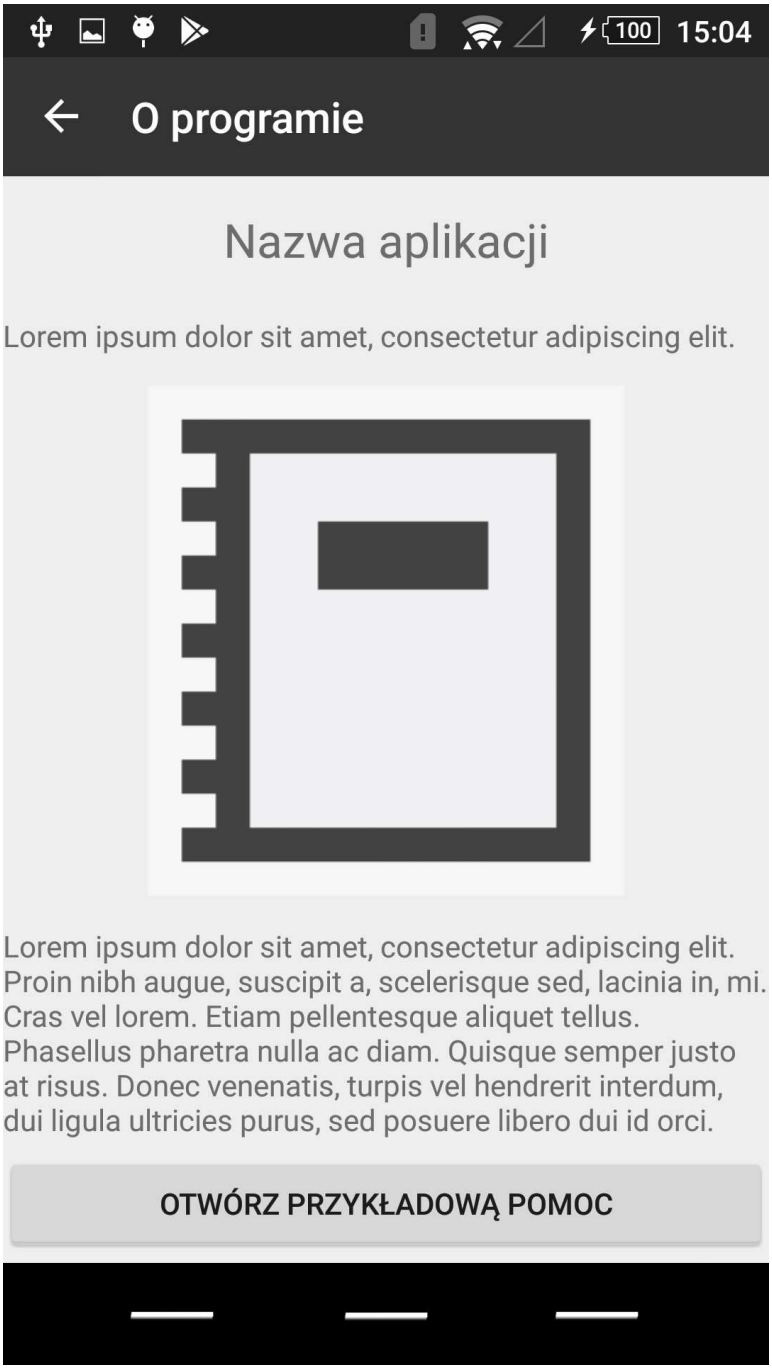
Jeśli dostęp do elementu jest chroniony blokadą rodzicielską, na ekranie pojawi się okienko monitujące o podanie kodu PIN. Kod PIN jest ustalany podczas tworzenia pliku konfiguracyjnego.



Rysunek 5: Okno odblokowania blokady rodzicielskiej

W przypadku podania błędnego PINu aplikacja wyświetla monit *Podano niewłaściwy kod PIN*.

Okienko pomocy i informacji o programie pozwala na umieszczenie w nim nazwy aplikacji, dystrybutora, danych kontaktowych, ikony programu oraz odesłania do materiałów z instrukcją obsługi.



Rysunek 6: Przykładowe okienko Pomocy i informacji o programie