Politechnika Koszalińska

Wydział Elektroniki i Informatyki

Sprawozdanie do projektu z przedmiotu:

„Zastosowania Sztucznej Inteligencji”

Semestr V, Studia Stacjonarne

Kierunek: Informatyka

Rok akademicki 2016/2017

Temat projektu: Aplikacja symulująca działanie sterownika kotła C.O. przy użyciu logiki rozmytej

Projekt wykonali:

Robert Mucha (nr indeksu U-10459)

Albert Mika (nr indeksu U-10456)

Data: 09.11.2016 r.

1. Wprowadzenie

Poniżej opisany projekt ma na celu zasymulowanie działania sterownika pieca C.O. przy uwzględnieniu współczynników takich jak:

- pogoda [temperatura] na zewnątrz ogrzewanego budynku
- temperatura wewnątrz pomieszczenia
- godzin pracy (pór dnia w jakich sterownik ma pracować)
- wielkości pomieszczenia, powierzchni ścian i okien

Dzięki zastosowaniu logiki rozmytej jesteśmy w stanie powyższe parametry przekształcić z naszej ludzkiej formy do postaci czytelnej dla komputera co sprawi, że symulacja będzie jak najbardziej zbliżona do 'naturalnego' procesu palenia, ponieważ w tym przypadku komputer [nasza aplikacja] będzie tak jak człowiek reagowała i analizowana na różne czynniki stąd też w przypadku sztucznej inteligencji wybór padł na logikę rozmytą - jest ona najlepsza w zastosowaniu dla problemów decyzyjnych.

2. Opis rozwiązywanego problemu

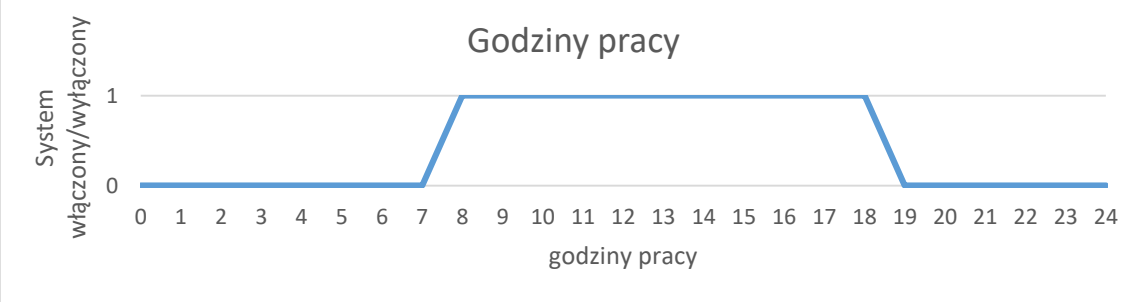
Rozwiązywanym problemem jest: dostosowanie odpowiedniej mocy grzewczej pieca przez sterownik w taki sposób aby jak najszybciej osiągnąć i utrzymać zadaną temperaturę przez użytkownika przy określonych warunkach/czynnikach jak i reagować na zmiany [zmienne wejściowe] które wprowadzi użytkownik na początku lub w trakcie jego działania. Na czynnik ogrzewania wpływa: wielkość ogrzewanego pomieszczenia, temperatura na zewnątrz, godziny pracy, temperatura zadana, obecna temperatura w pomieszczeniu natomiast docelowym parametrem jest temperatura pieca na podstawie której liczymy przyrost temperatury w ogrzewanym pomieszczeniu.

3. Opis techniki sztucznej inteligencji użytej w tworzonym projekcie

W projekcie istnieją trzy czynniki bezpośrednio zmieniające temperaturę: temperatura na zewnątrz budynku, grzanie pieca i chłodzenie przez klimatyzację. W niniejszej aplikacji pogoda jest czynnikiem losowym i nie zostaje ona poddana rozmyciu, pozostałe dwa czynniki są w pełni „poddane logice rozmytej", posiadają zmienne wejściowe i wyjściowe, przesłanki, wnioski i są wyostrzane. Klimatyzacja działa na zasadzie takiej jak piec z tą różnicą, że piec powoduje wzrost temperatury a klimatyzacja ją zmniejsza w przypadku przekroczenia limitu temperatury docelowej, dlatego też oba elementy korzystają z tych samych zbiorów zmiennych.

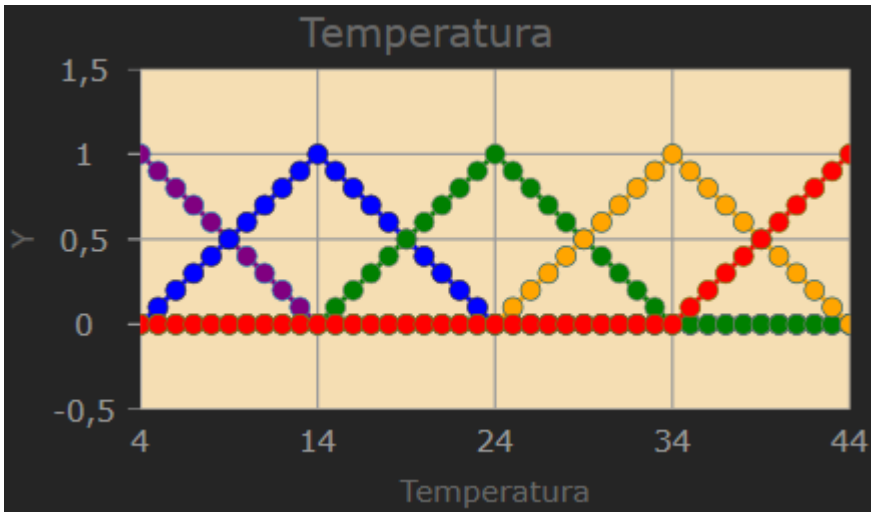
Zmienne lingwistyczne:

Godziny pracy – zmienna wejściowa tworząca przesłankę 'tak' lub 'nie', jest to klasyczny zbiór prostokątny. Jeżeli z tego zbioru otrzymamy przesłankę „Tak" to piec lub klimatyzacja może zostać włączona w przeciwnym wypadku systemy są wyłączone. Godziny pracy są wartością ustalaną przez użytkownika (domyślnie program ustawia godziny pracy od 05:00 do 10:00).



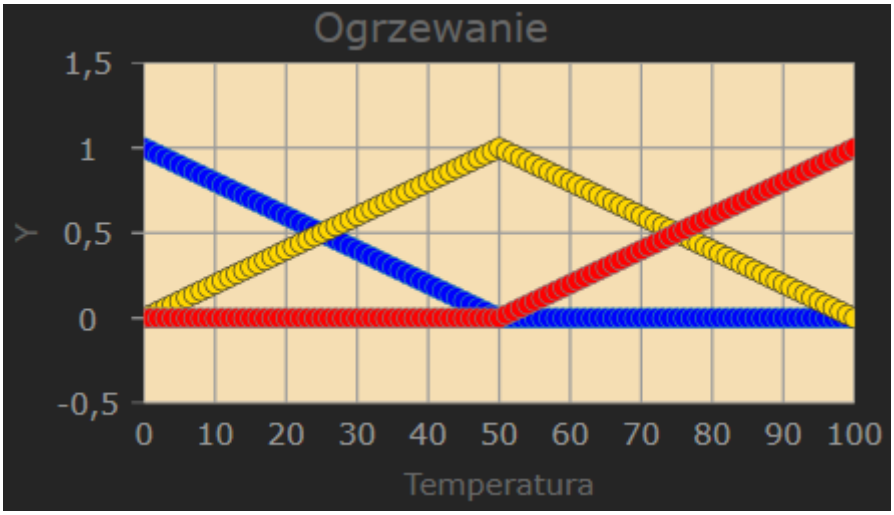
*Rysunek 1 graficzna prezentacja klasycznego zbioru dla godzin pracy**

Temperatura zadana będąca jedną ze zmiennych wejściowych. Określana przez użytkownika w zakresie od 10°C do 40°C (domyślna wartość: 20 °C). Poddawana rozmyciu na zbiory [temperatura] „Bardzo niska”, „Niska”, „Średnia” (czyli ta w której środek ‘trójkąta’ to wartość zadana), „Wysoka”, „Bardzo wysoka”. Na podstawie przesłanki z tej zmiennej lingwistycznej dowiadujemy się do jakiej temperatury nasz piec może maksymalnie ogrzać pomieszczenie. Ustanawia górny limit pieca i dolny limit dla klimatyzacji.



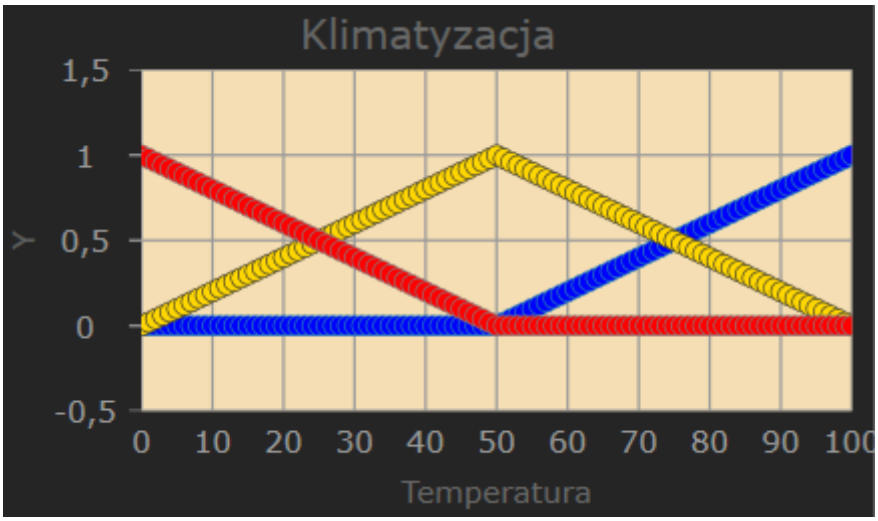
Rysunek 2 graficzna prezentacja zbioru rozmytego zadanej temperatury

Temperatura pieca jest jedną z dwóch wyjściowych zmiennych. Wartość maksymalna temperatury pieca może wynosić 70°C. Zmienna jest rozmywana na zbiory „Wysoka”, „Średnia”, „Niska”. Wartość dynamicznie zmienia się wraz z upływem czasu a wpływ na jej zmiany ma głównie wartość zmiennych „temperatura zadana” oraz „[obecna] temperatura pomieszczenia”



Rysunek 3 graficzna prezentacja zbioru rozmytego temperatury pieca

Klimatyzacja jest drugą z wyjściowych zmiennych. Minimalna wartość do której klimatyzacja może zbić temperaturę to 10°C. Zmienna jest rozmywana na zbiory „Niska”, ”Średnia”, ”Wysoka”. Wartość dynamicznie zmienia się wraz z upływem czasu a wpływ na jej zmiany ma głównie wartość zmiennych „temperatura zadana” oraz „[obecna] temperatura pomieszczenia”



Rysunek 4 graficzna prezentacja zbioru rozmytego klimatyzacji

Gdy otrzymamy wyjścia zmiennych lingwistycznych *Klimatyzacja* oraz *Temperatura pieca* to metodą środka ciężkości dla każdej z osobna obliczamy konkretną wartość która zostanie zadana dla urządzenia czyli dla pieca będzie to ‘nowa docelowa’ temperatura pieca a dla klimatyzacji temperatura do ustawienia na regulatorze.

Na koniec posiadając konkretne wartości po wyostrzeniu metodą środka ciężkości liczony jest współczynnik przyrostu temperatury w pomieszczeniu, określony wzorem:

$$\Delta t = \frac{temperatura\ pieca}{objętość\ pomieszczenia} + temperatura\ pogody - klimatyzacja$$

$$objętość\ pom. = \left(wysokość\ ścian * \sqrt{pow.pom.} - \frac{pow.okien}{2}\right) * \sqrt{pow.pom.} * \frac{1}{10}$$

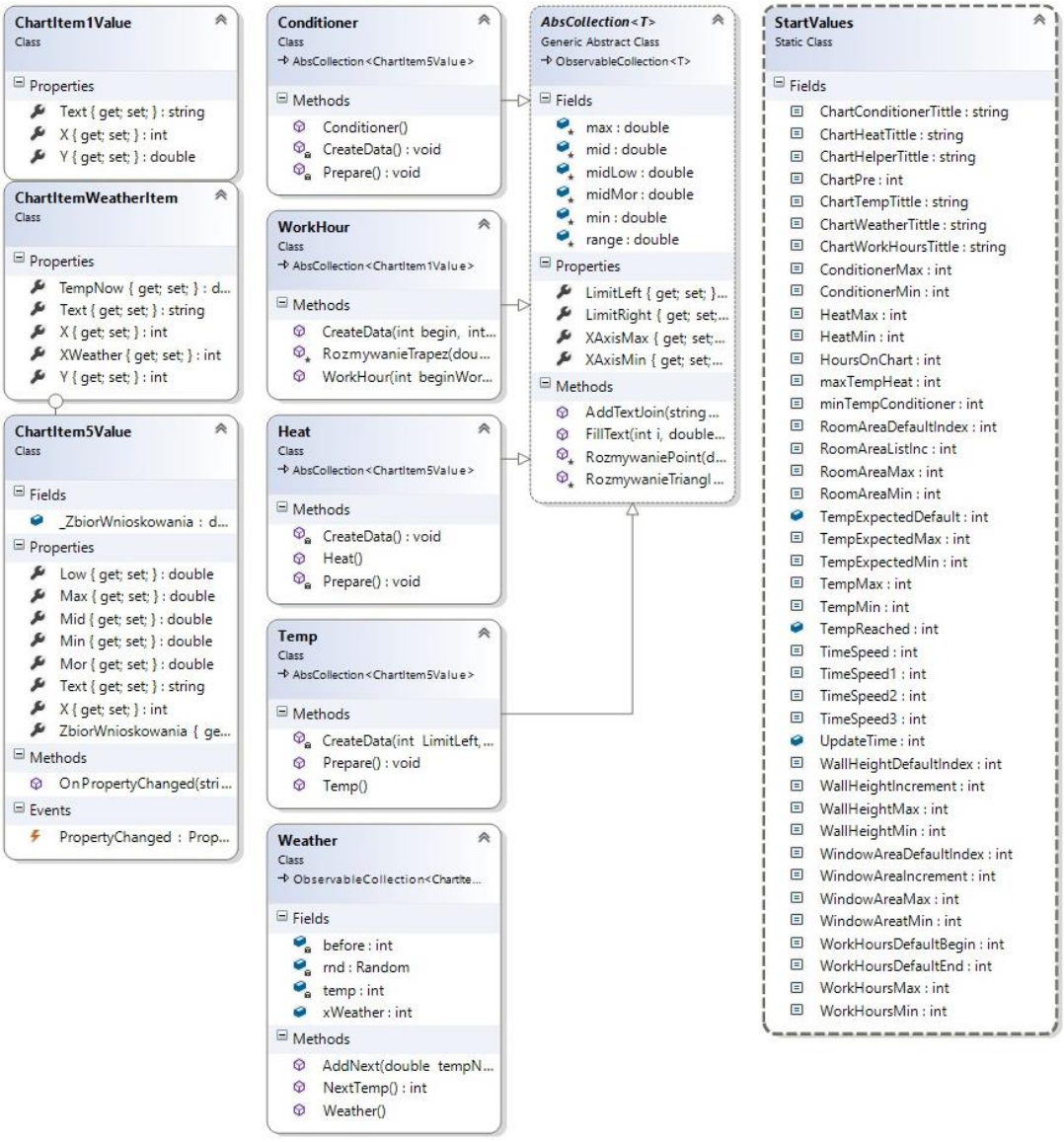
$$temperatura\ pogody = \log_{10}(|(temp.obecna + 1) - temp.na\ zewnątrz|)$$

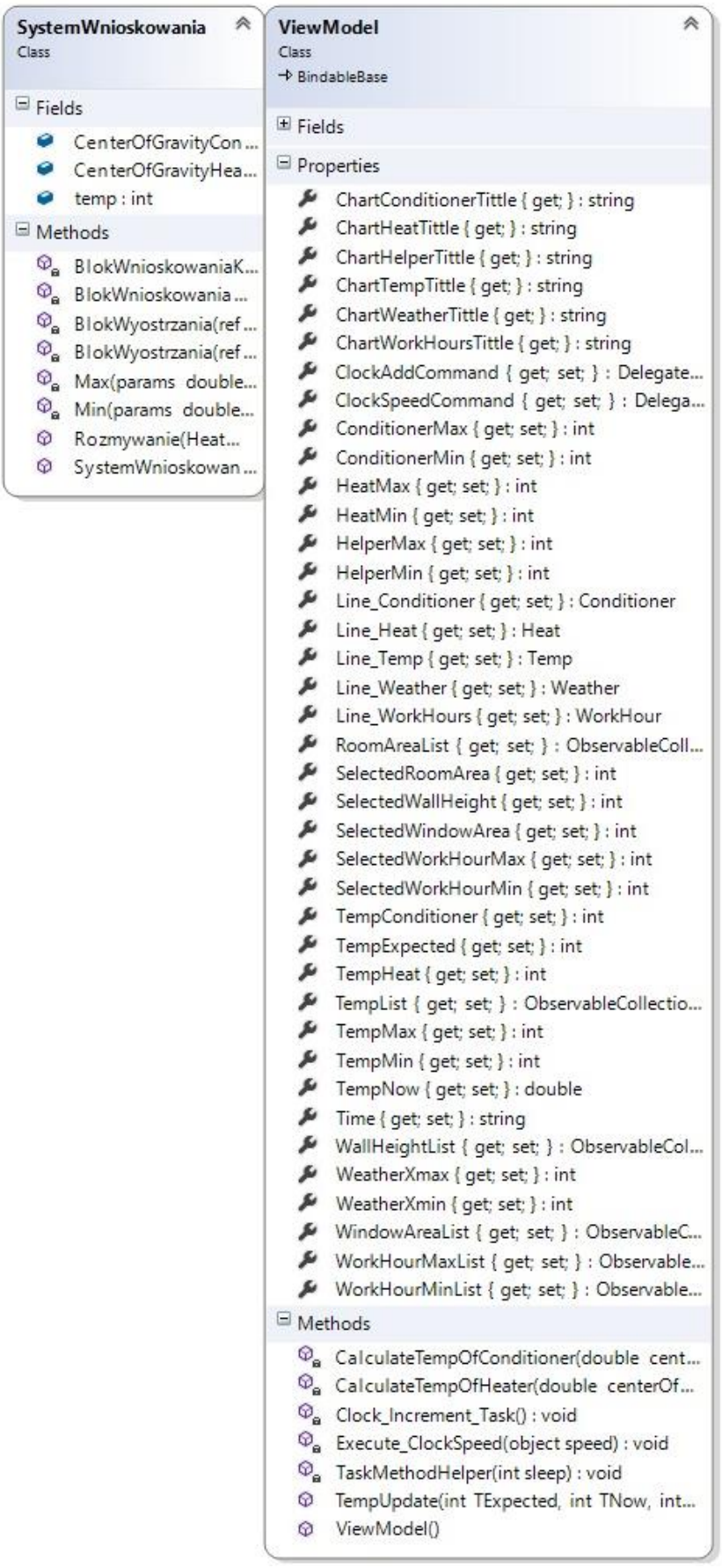
Tabela 1 skrócona lista reguł

Godziny pracy	Temperatura obecna pomieszczenia	Temperatura pieca	Klimatyzacja
NIE	*(brak wpływu)*	wyłączony piec	wyłączona
TAK	niska	wysoka	wyłączona
TAK	średnia	średnia	wyłączona
TAK	wysoka	niska/wyłączony piec	wysoka/średnia/niska

4. Diagram przypadków użycia utworzonej aplikacji

5. Diagram klas utworzonej aplikacji




SystemWnioskowania
Class
Fields
CenterOfGravityCon...
CenterOfGravityHea...
temp : int
Methods
BlokWnioskowaniaK...
BlokWnioskowania...
BlokWyostrzania(ref...
BlokWyostrzania(ref...
Max(params double...
Min(params double...
Rozmywanie(Heat...
SystemWnioskowan...
ViewModel
Class
BindableBase
Fields
Properties
ChartConditionerTittle { get; } : string
ChartHeatTittle { get; } : string
ChartHelperTittle { get; } : string
ChartTempTittle { get; } : string
ChartWeatherTittle { get; } : string
ChartWorkHoursTittle { get; } : string
ClockAddCommand { get; set; } : Delegate...
ClockSpeedCommand { get; set; } : Delega...
ConditionerMax { get; set; } : int
ConditionerMin { get; set; } : int
HeatMax { get; set; } : int
HeatMin { get; set; } : int
HelperMax { get; set; } : int
HelperMin { get; set; } : int
Line_Conditioner { get; set; } : Conditioner
Line_Heat { get; set; } : Heat
Line_Temp { get; set; } : Temp
Line_Weather { get; set; } : Weather
Line_WorkHours { get; set; } : WorkHour
RoomAreaList { get; set; } : ObservableColl...
SelectedRoomArea { get; set; } : int
SelectedWallHeight { get; set; } : int
SelectedWindowArea { get; set; } : int
SelectedWorkHourMax { get; set; } : int
SelectedWorkHourMin { get; set; } : int
TempConditioner { get; set; } : int
TempExpected { get; set; } : int
TempHeat { get; set; } : int
TempList { get; set; } : ObservableCollectio...
TempMax { get; set; } : int
TempMin { get; set; } : int
TempNow { get; set; } : double
Time { get; set; } : string
WallHeightList { get; set; } : ObservableCol...
WeatherXmax { get; set; } : int
WeatherXmin { get; set; } : int
WindowAreaList { get; set; } : ObservableC...
WorkHourMaxList { get; set; } : Observable...
WorkHourMinList { get; set; } : Observable...
Methods
CalculateTempOfConditioner(double cent...
CalculateTempOfHeater(double centerOf...
Clock_Increment_Task() : void
Execute_ClockSpeed(object speed) : void
TaskMethodHelper(int sleep) : void
TempUpdate(int TExpected, int TNow, int...
ViewModel()

```
public static class StartValues{}
```
Klasa zawierająca zbiór wszystkich zmiennych/stałych

```
class SystemWnioskowania
{
```
Klasa odpowiedzialna za rozmywanie, wnioskowanie i wyostrzanie wartości

```
    public void Rozmywanie( Heat Out1, Conditioner Out2, Temp InChart1, int value1,
    WorkHour InChart2,int index2){}
```
Metoda dokonująca rozmycia dla: temperatury pieca, klimatyzacji, temperatury, godzin pracy

```
    private void BlokWnioskowaniaOgrzewania(ref Heat Out, ChartItem5Value in1,
    ChartItem1Value in2){}
```
Metoda wnioskująca dla pieca

```
    private void BlokWyostrzania(ref Heat Out){}
```
Metoda dokonująca wyostrzania metodą środka ciężkości dla temperatury pieca

```
    private void BlokWnioskowaniaKlimatyzacji(ref Conditioner Out, ChartItem5Value in1,
    ChartItem1Value in2){}
```
Metoda dokonująca wnioskowania dla klimatyzacji

```
    private void BlokWyostrzania(ref Conditioner Out){}
```
Metoda dokonująca wyostrzania metodą środka ciężkości dla klimatyzacji

```
    private double Min(params double[] Values){}
```
Metoda wybierająca najmniejszą wartość z całej tablicy która jest podana jako argument

```
    private double Max(params double[] Values){}
```
Metoda wybierająca maksymalną wartość z całej tablicy która jest podana jako argument

```
}

class Conditioner:AbsCollection<ChartItem5Value>
{
    private void Prepare(){}
```
Metoda określająca zakres wartości osi X dla wykresu klimatyzacji

```
    private void CreateData(){}
```
Metoda generująca dane do rysowania wykresu wnioskowania dla klimatyzacji

```
}

class Heat: AbsCollection<ChartItem5Value>
{
    private void Prepare(){}
```
Metoda określająca zakres wartości osi X dla wykresu temperatury pieca

```
    private void CreateData(){}
```
Metoda generująca dane do rysowania wykresu wnioskowania dla temperatury pieca

```
}

class Temp : AbsCollection<ChartItem5Value>
{
    public void Prepare(){}
```
Metoda określająca zakres wartości osi X dla wykresu temperatury

```
    private void CreateData(int LimitLeft, int LimitRight){}
```
Metoda generująca dane do rysowania wykresu wnioskowania dla temperatury

```
}
```

```
class Weather:ObservableCollection<ChartItemWeatherItem>
{
    public int NextTemp(){}
```
Metoda generująca „kolejny stopień temperatury” dla pogody panującej na zewnątrz

```
    public void AddNext(double tempNowBuildidn,int tempNowWeather){}
```
Metoda dynamicznie aktualizująca wartości wyświetlane na wykresie zawierającym zarówno temperaturę pogody jak i wewnątrz pomieszczenia

```
}

class WorkHour : AbsCollection<ChartItem1Value>
{
    public void CreateData(int begin, int end){}
```
Metoda przydzielająca „bool’e” dla godzin pracy systemu [przydziela 0 lub 1]

```
    protected double RozmywanieTrapez(double x, double left, double right, double range)
```
Metoda generująca dane do wykresu godzin pracy

```
}

class ViewModel : BindableBase
{
    Główna klasa w której są zdefiniowane metody określające szybkość działania programu,
    oraz metody wywołujące inne metody liczące wszelkie wartości

    private void Execute_ClockSpeed(object speed){}
```
Metoda ustawiająca szybkość działania w zależności od kliknięgo przycisku

```
    private void Clock_Increment_Task(){}
```
Metoda wywołująca inne metody zgodnie z zadanym interwałem

```
    private void TaskMethodHelper(int sleep){}
```
Metoda symulująca czas w naszej aplikacji – odświeża wartości dla wyostrzania, przesuwania wykresów, wyliczania zmian temperatur (w pomieszczeniu i na zewnątrz), losowania kolejnej temperatury dla pogody

```
    private void CalculateTempOfHeater(double centerOfGravity){}
```
Metoda obliczająca temperaturę pieca uwzględniając maksymalną temperaturę pieca, wyliczaną wartość z wyostrzania metodą środka ciężkości i w oparciu o temperaturę docelową

```
    private void CalculateTempOfConditioner(double centerOfGravity){}
```
Metoda obliczająca temperaturę klimatyzacji uwzględniając możliwą minimalną temperaturę klimatyzatora, wyliczaną wartość z wyostrzania metodą środka ciężkości i w oparciu o temperaturę docelową

```
    public void TempUpdate(int TExpected, int TNow, int THeat, int TConditioner, int
    TWeather){}
```
Obliczanie przyrostu temperatury [sposób obliczania omówiony w pkt. 3]

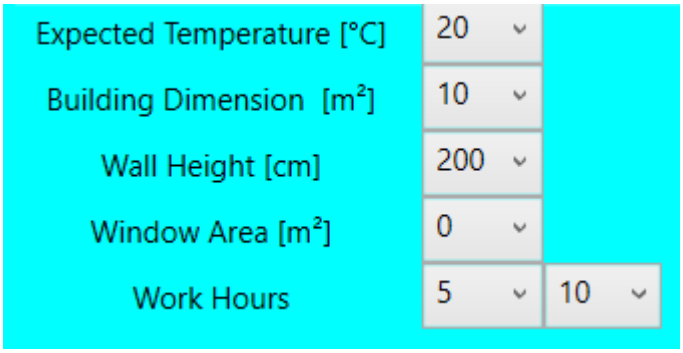
```
}
```

6. Opis utworzonej aplikacji – dokumentacja użytkownika

Zmiana parametrów środowiska:

Użytkownik po uruchomieniu aplikacji za pomocą list rozwijanych może zmienić wartości takich parametrów jak:

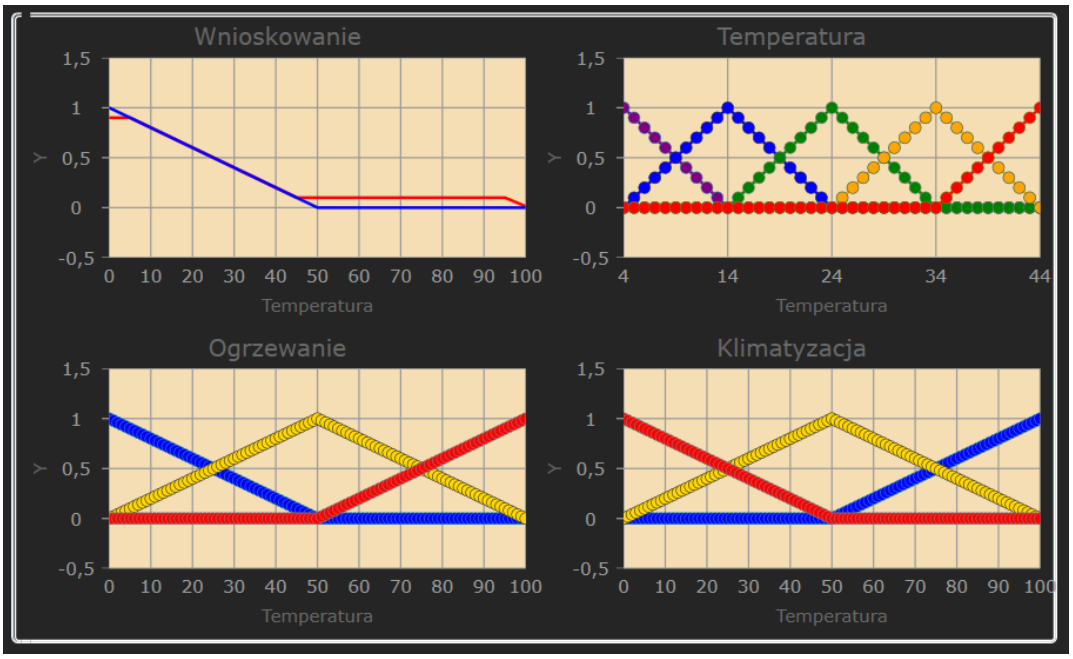
- temperatura docelowa
- powierzchnia grzanego budynku
- wysokość ścian
- powierzchnia okien
- godziny pracy pieca i klimatyzacji



Rysunek 5 miejsce zmiany parametrów programu

Powiększenie wybranego obszaru wykresu, odczyt wartości:

W programie po prawej stronie znajduje się sekcja wykresów prezentujących między innymi rozmycie danych wejściowych. Użytkownik chcąc dokonać ich dokładniejszej analizy może na wybranym wykresie zaznaczyć obszar lewym przyciskiem myszy – spowoduje to przybliżenie (powiększenie) wybranego obszaru. W celu odczytania wartości wystarczy najechać na linie wykresu a pojawi się ‘chmurka’ z danymi dla danego punktu.



Rysunek 6 wykresy obrazujące wartości przetwarzane przez program

Zmiana tempa upływu czasu w programie:

Można w dowolnym momencie dokonać zmiany szybkości działania symulacji lub ją zatrzymać poprzez kliknięcie na odpowiedni przycisk.

Rysunek 7 przyciski regulujące szybkość przebiegu symulacji

7. Podsumowanie

Wyżej opisany projekt pozwolił nam wykorzystać w praktyce techniki inteligencji obliczeniowej. Dowiedzieliśmy się w jaki sposób można dokonać jej programowej implementacji oraz jak bardzo pomaga ona ‘zrozumieć komputerowi’ niektóre kwestie związane z funkcjonowaniem przedmiotów nas otaczających. Najważniejszym i głównym punktem podsumowania jest fakt, że dzięki logice rozmytej z powodzeniem udało nam się zasymulować pracę prawdziwego pieca co było głównym celem projektu – bez ‘ingerencji’ inteligencji obliczeniowej aplikacja posiadałaby dużo bardziej skomplikowany i prawdopodobnie nieoptymalny sposób implementacji. Aplikacja może zostać wykorzystana w prawdziwym sterowniku pieca. Obecny program można rozbudować o obsługę czujników temperatury np.: z Arduino aby pobierać wartość z czujnika w pomieszczeniu i na zewnątrz w celu zwiększenia realizmu projektu.