НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмна система адаптивного управління пристроями за технологією Smart Home»

Виконав:
студент IV курсу, групи КП-61
Коломієць Денис Дмитрович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент
Сулема Євгенія Станіславівна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри ПМА, к.т.н., доцент
Сирота Сергій Вікторович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем

«ЗАТВЕРДЖУЮ»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2019 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Коломійцю Денису Дмитровичу

1. Тема проєкту «Програмна система адаптивного управління пристроями за технологією Smart Home», керівник проєкту Сулема Євгенія Станіславівна, к.т.н., доцент, затверджені наказом по університету від «25» травня 2020 р. № 1181-с.

2. Термін подання студентом проєкту «15» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
- огляд існуючих рішень;
- обгрунтування вибору засобів реалізації;
- опис розроблених алгоритмів та підпрограм;
- аналіз розробленого рішення.

5. Перелік обов'язкового графічного матеріалу:
- схема запису даних (креслення);
- діаграма прецедентів рішення (креслення);
- схема компонентів віконної частини (плакат);
- схема архітектури розробленого рішення (плакат).

6. Консультанти:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент, к.т.н.		

7. Дата видачі завдання «31» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою роботи	16.11.2019	
2.	Розроблення та узгодження технічного завдання	05.12.2019	
3.	Розроблення структури системи	20.12.2019	
4.	Розроблення дизайну вікон та графічних елементів	12.01.2020	
5.	Програмна реалізація графічного інтерфейсу	15.02.2020	
6.	Програмна реалізація модулю контролю	01.03.2020	
7.	Розробка набору жестів	29.03.2020	
8.	Тестування системи	11.04.2020	
9.	Підготовка матеріалів текстової частини проєкту	10.05.2020	
10.	Підготовка матеріалів графічної частини проєкту	25.05.2020	
11.	Оформлення технічної документації проєкту	30.05.2020	

Студент Денис КОЛОМІЄЦЬ

Керівник проєкту Євгенія СУЛЕМА

АНОТАЦІЯ

Даний дипломний проєкт присвячений розробці адаптивного десктопного програмного додатку, що дозволяє контролювати прилади у розумному будинку за допомогою жестів.

У роботі виконано порівняльний аналіз існуючих засобів контролю приладів розумного будинку і найчастіше використовувані інтерфейси користувача, обґрунтовано вибір технологій та допоміжних бібліотек для реалізації інтерфейсу користувача і використання приладів зчитування жестів. Розроблений десктопний програмний додаток надає можливість користувачу налаштовувати набори жестів, які будуть використані для контролю відповідних приладів, і надає алгоритмічну реалізацію зчитування відповідних жестів. Результатом такого контролю є зміна показників і стану приладів, що відображається користувачу через зручний графічний інтерфейс.

У даному дипломному проєкті розроблено та досліджено архітектуру системи зв’язку засобів контролю і приладів розумного будинку, архітектуру користувацького інтерфейсу та алгоритми зчитування жестів.

ABSTRACT

This diploma project deals with the development of an adaptive desktop software application that allows controlling devices in a smart home with gestures.

The paper performs a comparative analysis of the existing means of control of smart home devices and the most frequently used user interfaces, substantiates the choice of technologies and auxiliary libraries for the implementation of the user interface and the use of gesture readers. The developed desktop software application allows the user to customize the sets of gestures that will be used to control the respective devices and provides an algorithmic implementation of reading the corresponding gestures. The result of such control is a change in the performance and status of devices, which is displayed to the user through a user-friendly graphical interface.

Architecture of the communication system of control devices and devices of a smart home, the architecture of the user interface and gesture reading algorithms were researched and developed in this diploma project.

ДП.045440-01-90 Програмна система адаптивного управління пристроями за технологією Smart Home. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Програмна система	5	
	Адаптивного управління		
	пристроями за технологією		
	Smart Home. Технічне		
	завдання		
ДП.045440-03-81	Програмна система	50	
	Адаптивного управління		
	пристроями за технологією		
	Smart Home. Технічне		
	завдання. Програмна		
	система		
ДП.045440-04-51	Програмна система	4	
	Адаптивного управління		
	пристроями за технологією		
	Smart Home. Програма		
	та методика тестування		
ДП.045440-05-34	Програмна система	22	
	Адаптивного управління		
	пристроями за технологією		
	Smart Home. Керівництво		
	користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Програмна система	1	
	Адаптивного управління		
	пристроями за технологією		
	Smart Home. Загальна		
	архітектура системи.		
	Діаграма використання		
	Системи.		
ДП.045440-07-99	Програмна система	1	
	Адаптивного управління		
	пристроями за технологією		
	Smart Home. Загальна		
	архітектура системи.		
	Розроблена система.		
ДП.045440-08-98	Програмна система	1	
	Адаптивного управління		
	пристроями за технологією		
	Smart Home. Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2019 р.

ПРОГРАМНА СИСТЕМА АДАПТИВНОГО УПРАВЛІННЯ ПРИСТРОЯМИ ЗА ТЕХНОЛОГІЄЮ SMART HOME

Технічне завдання

ДП.045440-02-91

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Євгенія СУЛЕМА

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Денис КОЛОМІЄЦЬ

2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмна система адаптивного управління пристроями за технологією Smart Home.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості програмної системи адаптивного управління приладів у розумному будинку зі допомогою жестів із використанням приладу зчитування Kinect, розширюючи можливості взаємодії із будинками, що використовують технологію Smart Home.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Система має забезпечувати такі основні функції:

1. Реалізація архітектурного шаблону MVC, що забезпечує розбиття програмного коду, що відповідає за певний тип взаємодії із системою, на три різні модулі – Model, View, Control;
2. Реалізація архітектурного шаблону Strategy, що дозволяє швидко розширювати функціональні можливості за рахунок реалізації єдиної моделі взаємодії із певними фрагментами системи;
3. Функціональні можливості для конструювання набору жестів користувачем

4. Функціональні можливості для збереження та відтворення наборів жестів у форматі, що зрозумілий для людини.
5. Функціональні можливості для контролю приладів за допомогою користувацьких наборів жестів через прилад зчитування жестів.
6. Модуль тестування, що дозволяє перевіряти виконуваність і зручних розроблених жестів для користувача і розробника.
7. Графічний інтерфейс для взаємодії із конструктором наборів і модулем тестування.
8. Графічний інтерфейс для відображення результатів контролю і стану приладів при взаємодії із системою.

Розробку виконати за допомогою мов програмування C# з використанням засобів розробки вікон WPF і модулю розробки застосунків із використанням приладу зчитування жестів Kinect SDK 1.8

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1. Пояснювальна записка.
2. Програма та методика тестування.
3. Керівництво користувача.
4. Креслення.

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи 16.11.2019

Розроблення та узгодження технічного завдання 05.12.2019

Розроблення структури системи 20.12.2019

Розроблення дизайну дизайну вікон та графічних елементів...12.01.2020

Програмна реалізація графічного інтерфейсу 15.02.2020

Програмна реалізація модулю контролю.....01.03.2020

Розробка набору жестів 29.03.2020

Тестування системи 11.04.2020

Підготовка матеріалів текстової частини проекту 10.05.2020

Підготовка матеріалів графічної частини проекту 25.05.2020

Оформлення технічної документації проекту 30.05.2020

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2020 р.

ПРОГРАМНА СИСТЕМА АДАПТИВНОГО УПРАВЛІННЯ ПРИСТРОЯМИ ЗА ТЕХНОЛОГІЄЮ SMART HOME

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Євгенія СУЛЕМА

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Денис КОЛОМІЄЦЬ

2020

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Kinect – безконтактний сенсорний контролер, спочатку представлений для консолі Xbox 360, і значно пізніше для персональних комп'ютерів під керуванням ОС Windows. Розроблений фірмою Microsoft. Заснований на додаванні периферійного пристрою до грального консолі Xbox 360, Kinect дозволяє користувачеві взаємодіяти з нею без допомоги контактного контролера через усні команди, пози тіла, об'єкти або малюнки [1].

Leap motion – це розроблювана технологія, заснована на захоплення руху, для людино-комп’ютерної взаємодії. Технологія реалізована у однойменному пристрою [2].

ПЗ – програмне забезпечення

AJAX – (Asynchronous JavaScript And XML) – підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані. AJAX – один з компонентів концепції DHTML [3].

JSON – (англ. JavaScript Object Notation) – це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією) [4].

XML – (англ. Extensible Markup Language, скорочено XML) – запропонований консорціумом World Wide Web Consortium (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML. XML-документ складається із текстових знаків, і придатний до читання людиною [5].

SGML – (англ. Standard Generalized Markup Language, SGML) – метамова, за допомогою якої можна визначати мову розмітки для документів. SGML – нащадок розробленої 1960 року в IBM мови GML (Generalized Markup Language), яку не варто плутати з Geography Markup Language, яку розробляв Open GIS Consortium [6].

MVC – (Модель–представлення–контролер, англ. Model-view-controller, MVC) – архітектурний шаблон, який використовується під час проєктування та розробки програмного забезпечення [7].

WPF – (Windows Presentation Foundation) графічна (презентаційна) підсистема (аналог WinForms), яка починаючи з .NET Framework 3.0 в складі цієї платформи. Має пряме відношення до XAML. WPF разом з .NET Framework 3.0 вбудована в Windows Vista, а також доступна для установки в Windows XP Service Pack 2 і Windows Server 2003 [8].

XAML – (англ. eXtensible Application Markup Language) – декларативна мова розмітки. З точки зору моделі програмування .NET Framework мова XAML спрощує створення користувацького інтерфейсу для програми .NET Framework [9].

ВСТУП

Основна мета дипломної роботи – розбробка програмного забезпечення для використання у розумному будинку для контролю приладів за допомогою жестів через різні контролери руху, такі як – kinect та leap motion.

Інтеграція технологій мікроконтролерів для взаємодії із різними приладами у будинку набуває все більших масштабів, що робить взаємодію зі всіма приладами у будинку швидшою та більш інтуїтивною. Тому розробка різних способів взаємодії з ними – важлива і корисна задача для подальшого використання багатьма користувачами.

Програмне забезпечення розроблене в цій дипломній роботі надає можливості для контролю приладів у будинку, а саме:

- Контроль приладів, що мають два режими – включений і виключений;
- Контроль приладів із безперервними значеннями – термостат або рівень освітлення;
- Контроль приладів із складними режимами – декілька можливих налаштувань;
- Перемикання між контрольованими приладами;
- Налаштування певних жестів для використання різних приладів;
- Інтерфейс для зв’язування приладів і жестів.

Беручи до уваги всі зазначені компоненти, отримаємо зручну та адаптивну систему для контролю будинку жестами.

Етап тестування має бути виконаний із врахуванням особливостей взаємодії, тому дане ПЗ передбачає розробку модулю для тестування як при розробці, так і при налаштуванні перед використанням.

Цьому дипломному проєкті буде проаналізовано вже існуючі рішення для контролю приладів у будинку і розроблено останню версію продукту з урахуванням всіх мінусів існуючих розробок.

Дане ПЗ не потребує від користувача спеціальних навичок програмування, що робить його доступним для використання будь-ким.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ОБГРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЄКТУ

1.1. Загальний опис проблеми контролю приладів у розумному будинку

Розумний будинок допомагає використовувати різні прилади у будинку як одну зв’язану між собою систему, що надає додаткові можливості для вирішення більш складних задач у будинку і підвищити взаємну інтеграцію різних приладів між собою. Прикладами таких рішень є зв’язок між будильником і чайником для їх синхронізації – користувач може заварювати чай із водою певної температури як тільки він прокинувся, економлячи час на нагрівання води. Така ж інтеграція може відбуватися з бойлером для економії енергії на нагрівання води певну частину і піднімаючи температуру лише ближче до ранку.

Також такі системи надають можливість віддаленого контролю приладів, наприклад включення світла при довгих подорожах для того щоб відвадити зловмисників від будинку. Іншим прикладом є зручне відстежування відео з камер на території будинку зі смартфону або комп’ютера.

Отже загалом – система розумного будинку надає надзвичайно широкі можливості для контролю і інтеграції приладів у будь-якій оселі, які були підключені до певного контролера. Особливо така інтеграція несе цінність для розробників, що можуть відстежувати закономірності у поведінці користувачів вдома і виводити нові можливості взаємодії усіх компонентів системи.

Після винаходу винаходу мікроконтролерів, електроприлади стали більш уживаними через зниження собівартості. Таким чином, почали з’являтися прилади, що здатні були забезпечити більшу ефективність у побутовому використанні робочих і житлових приміщень. У 1984 році вперше був згаданий термін «розумний будинок» у постанові Американської Асоціації Housebuilders у [10].

У 1995 році компанія – розробник мови програмування Java [11] визначила як одне із призначень цієї мови розробку застосунків для інтеграції із мікроконтролерами через архітектурні рішення, пов’язані із віртуальною машиною JVM. Це дозволяло розробляти програми для будь-яких пристроїв, що мають встановлену віртуальну машину.

Основні якості розумного будинку:

- інтеграція багатьох приладів в одну систему, що дозволяє привнести складну логіку в управління будинком;
- велика кількість сучасних приладів, що постачаються з легко інтегрованими мікросхемами;
- невелика ціна рішень розумних будинків для ентузіастів, що дозволяє отримувати якісне програмне забезпечення за системою open source;
- зручність – кінцевий користувач швидко звикає і пристосовується до сучасних рішень у сфері контролю за будинком;
- фінансова вигода - в багатьох країнах існують певні обмеження на використання енергії в певний час на добу, тому прилади можуть бути запрограмовані відповідно до цих правил;
- підвищена безпека таких осель – відвада зловмисників через симуляцію повсякденної діяльності в оселі або відстежування обладнання з відео нагляду у будинку.

На основі трендів підвищення інтеграції технологій в усіх сферах повсякденної життєдіяльності можна зробити висновок, що подібні рішення будуть набувати популярності в майбутньому, тому розробка систем контролю за розумними будинками – важлива задача для різних розробників.

1.2. Аналіз існуючих рішень для даної задачі

Основні способами контролю різних приладів і взаємодія із розумним будинком представлені на ринку є:

- контроль за допомогою традиційних вбудованих інтерфейсів;
- контроль через смартфон або комп’ютер;
- взаємодія через голосового асистента.

Існують і інші можливості керування і взаємодії, але вони є менш популярними через відсутність зручних розроблених систем, високу вартість або потребу у додатковій кваліфікації користувачів.

Контроль за допомогою різних вбудованих інтерфейсів

Хоча концепція розумного будинку включає у себе розумний контроль і налаштування приладів, стандартний тип взаємодії все ще використовується у будинках разом із технологією «Smart Home». Наприклад, термостат хоча і може бути запрограмований на зчитування даних з багатьох датчиків у будинку, він також може бути контрольований за допомогою стандартного перемикача температури або пульта.

Переваги ПЗ:

- традиційний спосіб використання, до якого звикла більшість користувачів;
- головні технології для взаємодії є розробленими і протестованими великою кількістю користувачів;
- недорогі рішення, що є доступними будь-яким користувачам.

Недоліки ПЗ:

- відсутність загального інтерфейсу для контролю багатьох приладів як загальної системи;
- відсутність додаткової інтеграції і простих налаштувань логіки зв’язку між багатьма приладами;
- відсутність додаткових способів взаємодії з користувачем через інтернет.

Контроль через смартфон або комп'ютер

Будь-який користувач системи розумного будинку матиме доступ до свого смартфону або комп'ютера. Такі системи покращують зв'язок між користувачем і системою загалом, тому дозволяють йому контролювати усі прилади у будинку як знаходячись в оселі, так і поза її межами за допомогою мережі Інтернет. Також ці інтерфейси допомагають налаштовувати складну логіку при використанні усієї системи.

Переваги ПЗ**:**

- зв'язок з системою розумного будинку у віддаленому режимі за допомогою мережі Інтернет;
- налаштування складної логіки взаємодії між приладами;
- відстежування великої кількості показників для можливого подальшого аналізу.

Недоліки ПЗ**:**

- мала інтеграція з повсякденними задачами користувача, такими як віддалений швидкий контроль приладів, наприклад включення чайнику;
- потреба в довгій взаємодії для вилучення простої інформації, наприклад – температури в кімнаті.

Взаємодія через голосового асистента

Сучасні технологічні компанії, такі як Amazon, пропонують голосових асистентів для взаємодії з будинком. Такий інтерфейс – дуже зручний додаток до системи розумного будинку, оскільки користувач може контролювати усі прилади за допомогою голосу, виконуючи повсякденні справи. Також цей інтерфейс дуже швидко може сповіщати будь-яку інформацію у форматі голосових питання-відповіді. Додаткова перевага такого інтерфейсу – безпека, що набувається через розпізнавання лише певної групи користувачів.

Переваги ПЗ**:**

- швидка взаємодія з користувачем у будь-який час, що дозволяє виконувати функції деяких інших приладів, таких як органайзер або будильник;
- не прив’язаний до певного місця у будинку, тому його можна використовувати в межах усієї оселі;
- відсутність тактильної взаємодії, що дозволяє покращити якість життя для користувачів із вадами руху.

Недоліки ПЗ**:**

- прив’язка до певного голосу користувача, що призводить до погіршення взаємодії для користувачів у яких по певним причинам змінюється голос;
- не надає ніяких можливостей користувачам, які не можуть комунікувати за допомогою голосу;
- складний у локалізації, оскільки потребує довгої роботи великої команди висококваліфікованих професіоналів для адаптації під різні мови і діалекти, що можуть надзвичайно сильно відрізнятися за багатьма критеріями.

1.3. Актуальність розробки застосунку

У попередньому розділі із аналізом існуючих рішень наведено різні існуючи способи взаємодії із системою за технологією Smart Home [12] було виявлено, що існує потреба у додаткових способах взаємодії задля полегшення використання груп користувачів. У даному дипломному проєкті розроблювана програмна система призначена для управління розумним будинком із використанням приладів зчитування жестів. Також, прилади зчитування жестів, що використані у даній роботі, є достатньо розповсюдженими, роблячи дане ПЗ більш актуальним для великого пласту користувачів.

Отже, дане ПЗ призначене для використання разом із приладами розумного будинку в системах із наявними приладами. Дане ПЗ може бути інтегроване у існуючу систему контролю за технологією Smart Home.

1.4. Загальні вимоги до системи

В результаті проведеного дослідження можна сформулювати такі вимоги до програмного забезпечення для контролю приладів у розумному будинку за допомогою жестів:

- реалізація моделі MVC (Model View Control) для розмежування функціональні можливості системи на різні модулі: відображення результату контролю на View сегменті, зміна даних в результаті контролю на сегменті Model та контроль приладів на сегменті Control;
- функціональні можливості для перегляду результату контролю приладів;
- функціональні можливості контролю приладів за допомогою приладів зчитування жестів;
- ПЗ має включати сторінку для налаштування різних наборів жестів і різних типів взаємодії із наявними приладами розумного будинку;
- відтворення і зручне представлення результатів контролю приладів розумного будинку;
- ПЗ має включати можливість обирати різні зібрані набори, зберігати та відтворювати збережені набори для підвищення зручності користування декількома користувачами системи на різних пристроях;
- легке переключення між наборами жестів без надлишкової взаємодії, так само як і перенесення файлів збереження розкладок жестів.

Наведені вище вимоги формують рішення поставленої задачі на основі переваг та недоліків вже існуючих аналогів.

1.5. Вимоги до жестів

Функціональні можливості даної системи заснований на зчитуванні певного положення тіла користувача. Оскільки система розраховане на безперервне функціонування довгий проміжок часу для реєстрації жесту, важливо сформувати вимоги таким чином, щоб під час функціонування не виникало реєстрації жестів, що користувач виконував випадково. Іншим важливим пунктом є виконуваність жестів користувачем і відсутність надмірного фізичного навантаження для підвищення доступності більшій категорії користувачі. Усі жести, що розроблені таким чином, мають реєструватися абсолютно індивідуально так, щоб не викликати дію на реєстрацію при виконанні іншого жесту.

Виходячи з наведеного вище, можна сформувати такі вимоги для до розроблюваного набору жестів:

- простота жесту – більшість користувачів не залежно від їх фізичних можливостей можуть виконати будь який жест без надмірних перевантажень;
- відсутність колізій – усі жести, що будуть присутні у розробленому наборі мають бути виконуваними без реєстрації інших жестів;
- нетиповість – розроблені жести мають бути такими, що рідко виконуються при нормальній діяльності користувача.

1.6. Висновки до розділу

Розвиток технології Smart Home спричиняє зростаючий попит на інструменти взаємодії із усіма інтегрованими приладами, що включені у систему будь-якого розумного будинку. Із розвитком технології зростає кількість користувачів, що потребують персональних методів взаємодії із такою системою. Такі користувачі мають різні можливості, пов’язані із методами взаємодії із даною технологією. Таким чином, вони потребують

більш широкого вибору методів взаємодії для зручнішої взаємодії із системою Smart Home.

Розроблюване програмне забезпечення у вигляді десктопного додатку призначене для розширення можливостей взаємодії із приладами розумного будинку через реалізацію системи контролю показників таких приладів із використанням технологій зчитування жестів.

Існуючи методи надають широкі можливості взаємодії, але не покривають усіх потреб і не враховують певні можливі вади різних категорій користувачів.

Програмне забезпечення має бути легко інтегровне у вже існуючи системи контролю за рахунок архітектури, що дозволяє легко виділяти і використовувати програмні рішення, що були використані у даному дипломному проєкті.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору мови програмування

Оскільки для реалізації програмного забезпечення для контролю за приладами розумного будинку за допомогою жестів були обрані конкретні пристрої зчитування рухів, вибір мови програмування був заснований на зручному інструментарії розробки для приладів “Kinect” та “Leap Motion”.

Також важливим фактором при реалізації є зручність розробки інтерфейсу користувача для вибору набору жестів для контролю певних приладів.

При розробці даного проєкту необхідно враховувати, що мова програмування має бути достатньо швидкою для зручності використання графічного інтерфейсу користувачем та обробляти просторову інформацію з приладів відстежування рухів у реальному часі без затримок.

Огляд мови програмування C#

C# – розроблена компанією Microsoft у 2000 році об'єктно-орієнтована мова програмування. Вона є C-подібною, тому за синтаксисом вона найбільше схожа на інші об’єктно-орієнтовні мови, такі як *C++* та *Java*. Головним розробником є Андерс Хейлсберг [13].

C# є мовою із статичною типізацією, що підтримує різні парадигми програмування: процедурне, узагальнене, функціональне, подійно-орієнтоване.

Основною перевагою C# є розробка і офіційна підтримка продуктів компанії Microsoft, як компанії – розробника даної мови програмування. Таким чином, саме ця мова надає найбільш об’ємну документацію і приклади розробленого програмного коду для приладу зчитування *Kinect*.

Переваги *C#*:

- простота розробки графічного інтерфейсу;
- доступний та зрозумілий для читання код;

- наявність офіційних бібліотек для роботи із *kinect* і *leap motion.*

Основним недоліком при розробці на *C#* основним недоліком називають відсутність крос платформності, однак даний проєкт реалізується для єдиної операційної системи.

Огляд мови програмування Python

Python – високорівнева інтерпретовна мова програмування, що надає широкий інструментарій для розробки програм різних напрямків. Синтаксис заснований на комбінації відступів, що простіший для сприйняття і призводить до підвищення продуктивності. Python 1 був розроблений у 1991 році Гвидо Ван Россумом [15].

Python є мультипарадигмальною мовою програмування із динамічною типізацією. Підтримує структурне, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване програмування. *Python* інтегрований із пакетним менеджером pip, що дозволяє зручно використовувати сторонні бібліотеки. Мова *Python* є найпопулярнішою мовою для дослідницької розробки і широко використовується у веб-програмуванні.

Переваги *Python*:

- значна швидкість створення програмного забезпечення через простий синтаксис мови *Python*;
- наявність великої кількості відкритих бібліотек та постійний розвиток цієї мови програмування;
- легкість редагування коду та його зрозумілість;
- реалізовані бібліотеки для розробки на *kinect.*

Головним недоліком є низька швидкодія інтерпретовної мови, що може негативно відображатися на зручності використання користувачем програмного забезпечення, що оброблює просторові дані, приводячи до затримок і некоректної роботи комплексу.

Огляд мови програмування C++

C++ [14] – компільована С-подібна мова програмування, що розроблена у 1979 році Б'ярном Страуструпом. Підтримує декілька парадигм: об'єктно-орієнтовну, узагальнену та процедурну. При розробці даної мови основною ціллю була модифікація синтаксису С для використання класів.

Оскільки мова є компільованою і достатньо швидкою, вона використовується для розробки у сферах, де важливим є швидка обробка даних: при написанні драйверів, розробці багатопоточних систем, використовується як мова написання інтерпретаторів інших мов програмування, а також для обробки значних відео-потоків.

Переваги *C++*:

- швидкодія при роботі із великими потоками даних;
- бібліотека що надають простий інтерфейс для користувача.

Головний недолік розробки на *C++* – високі вимоги до здібностей програміста через відсутність автоматичного видалення невикористованих змінних і прямої роботи з пам'яттю.

Огляд мови програмування Java

Java – об’єктно-орієнтовна мова програмування, розроблена компанією Sun Microsystems (потім придбана Oracle) як мова для написання веб-аплетів. Програмний код написаний на Java транслюється у байт-код, що виконується JVM (*Java* Virtual Machine) віртуальною машиною, що виступає в якості інтерпретатора і передає інструкції обладнанню. Така архітектура реалізації програмно-машинної взаємодії дозволяє виконувати код *Java* у будь-якому апаратному середовищі, що дозволяє використовувати її для написання аплетів, простих програм для вбудованих процесорів у побутових приладах і на будь-якому сервері із встановленною віртуальною машиною. *Java* була розроблена у 1995 році Джемсом Гослінгом. *Java* є мультипарадигмальною мовою програмування, хоча найбільш популярною і розвиненою парадигмою в рамках даної мови є об’єктно-орієнтовне програмування [11].

Переваги Java:

- стабільність та наявність певної спільноти через довгий період застосування;
- є стандартом для корпоративних обчислювальних систем.

Серед недоліків використання мови програмування *Java* найчастіше згадується зниження продуктивності через використання концепції віртуальної машини.

Обрана мова програмування

Для розробки даного програмного забезпечення були проаналізовані мови і визначені такі недоліки:

- *Python* як інтерпретована мова програмування не надає достатньої швидкодії для роботи із потоком просторових даних і обробки інформації для розрахунку чисельних значень контролю приладів;
- *Java* також недостатньо продуктивна для зручної роботи із приладами і потоком відеоданих і просторової інформації і відгуку на жести;
- *C++* хоча і є найпродуктивнішою мовою із розглянутих, однак складність розробки не дозволяє використати її для розробки даного проєкту;
- *C#* маючи суттєвий недолік як не кросплатформна мова розробки, що є несуттєво для даного проєкту. Також маючи офіційну документацію і бібліотеки для розробки пов'язаних із *kinect*.

Проаналізувавши основні мови із бібліотеками для десктопної розробки на *kinect* був зроблений вибір на користь мови C# через зручність розробки, високу продуктивність і можливість дизайну простого користувацького інтерфейсу.

2.2. Обґрунтування вибору способу запису даних

Дане програмне забезпечення не передбачає збереження великої кількості даних, тому для нормального функціонування вистачить простого

файлу із збереженими налаштуваннями. Однак такі налаштування мають бути структурованими для зручності розробки і можливості доступу до них.

Збереження інформації про обрані розкладки жестів у базі даних є неактуальним, оскільки такі дані є простими і малими за об’ємом (при збереженні у файлі така розкладка не перевищуватиме двох екранів прокрутки). Таким чином, для реалізації буде достатньо простого збереження розкладки у файлі для простого доступу, так само як і зберегти таку розкладку для перенесення на інший пристрій.

Важливо зауважити, що такий файл не передбачає ніякої персональної інформації про користувача програми, таким чином отримання несанкціонованого доступу до файлу не несе загрозу для користувача.

Збереження у форматі JSON

це текстовий формат обміну даними між комп'ютерами. *JSON* базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передачі структурованої інформації через мережу (завдяки процесу, що називають серіалізацією) [4].

Розробив і популяризував формат Дуглас Крокфорд.

JSON знайшов своє головне призначення в написанні веб-програм, а саме при використанні технології AJAX. *JSON*, що використовується в AJAX, виступає як заміна XML (використовується в AJAX) під час асинхронної передачі структурованої інформації між клієнтом та сервером. При цьому перевагою *JSON* перед XML є те, що він дозволяє складні структури в атрибутах, займає менше місця і прямо інтерпретується за допомогою JavaScript в об'єкти.

Збереження у форматі XML

запропонований консорціумом World Wide Web Consortium (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною

підмножиною мови розмітки SGML. *XML*-документ складається із текстових знаків, і придатний до читання людиною [5].

Стандарт *XML* визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Ці обмеження описуються мовами схем

Обрана нотація

Оскільки аналізований проєкт не потребує методів пошуку у системі запису і нотації, а також обраною мовою програмування є C#, більш зручною нотацією для використання буде *XML*.

2.3. Висновки до розділу

У даному розділу був виконаний збір вимог до технологій розробки проєкту. Мова програмування і спосіб збереження даних були розглянуті з огляду на розроблюване програмне забезпечення у вигляді декстопного програмного застосунку. Метод аналізу – розгляд основним технологій розробки аналогічних застосунків, таких мов програмування, як *Python, C#, C++, Java*, а також способів збереження структурованих даних, як мови розмітки *XML* та *JSON*.

Для розробки була обрана мова програмування *C#*. Дана мова не надає невисокий степінь кросплатформності, але даний недолік не впливає на реалізацію даного контректного проєкту. Дана мова дає широкі можливості для розробки із використання приладу зчитування жестів *Kinect*, оскільки є реалізовані офіційні бібліотеки для реалізації застосунків, широку документацію і наглядні приклади. Із цього випливає зручність розробки, що позитивно впливає на швидкість і якість розробленого програмного забезпечення.

Як спосіб збереження даних була обрана мова розмітки *XML* через простоту зв’язку із обраною мовою програмування і простий доступ до збережених даних.

3. РОЗРОБЛЕННЯ ПРОГРАМНОЇ ЧАСТИНИ СИСТЕМИ КОНТРОЛЮ ПРИЛАДІВ РОЗУМНОГО БУДИНКУ ЗА ДОПОМОГОЮ ЖЕСТІВ

3.1. Загальний опис системи

Розроблювана програмна частина системи контрою приладів у розумному будинку має забезпечувати можливість контролю наборів жестів і вибору певних приладів для контролю певного приладу, надаючи можливості для широкого пристосування і зручного управління користувачем.

У системі передбачено три контрольованих прилади:

- чайник;
- термостат;
- світло.

Дані прилади можуть бути контрольовані із приладу зчитування жестів kinect. Користувач матиме можливість налаштування конкретного приладу і його зчитування із конкретного пристрою. Виходячи з цього, користувач матиме можливість поставити у відповідність до приладу зчитування певний прилад розумного будинку.

Чайник

Включення чайнику – проста реакція на певний жест користувача. У системі є певний набір запрограмованих жестів для розпізнавання і включення чайнику для обох приладів як фіксацію конкретного положення руки і подальшого включення приладу.

Термостат

Термостат контролює температуру у будинку або у кімнаті. Оскільки величина температури є безперервною, прилади мають фіксувати зміну положення кінцівки для відповідної зміни температури. Прилад має попередньо реєструвати жест, що відповідає за перехід у режим зміни температури на термостаті. Додатковою опцією для вибору користувача має

бути можливість переключення температури до максимальної та мінімальної температури на термостаті через реєстрацію одного жесту.

Світло

У сучасних будинках світло може бути переключене як із двох положень (включене і виключене) так і як безперервну величину для регуляції освітлення у кімнаті. Тому користувач матиме можливість обрати один із двох режимів контролю світла – переключення із двох можливих положень або зміна рівня освітлення у кімнаті. Зміна рівня освітлення відбуватиметься після першої реєстрації жесту для переходу у режим змінного освітлення.

Кожний контрольований прилад може бути призначений на відповідний прилад відстежування жестів.

Файл із збереженою конфігурацією містить збережену при попередньому використанні налаштування користувача. Кожний файл має унікальне ім’я, відповідною до назви збереженої комбінації. Файли мають текстовий формат із розміткою XML.

Діаграму загального використання системи можна побачити на рис. 1.

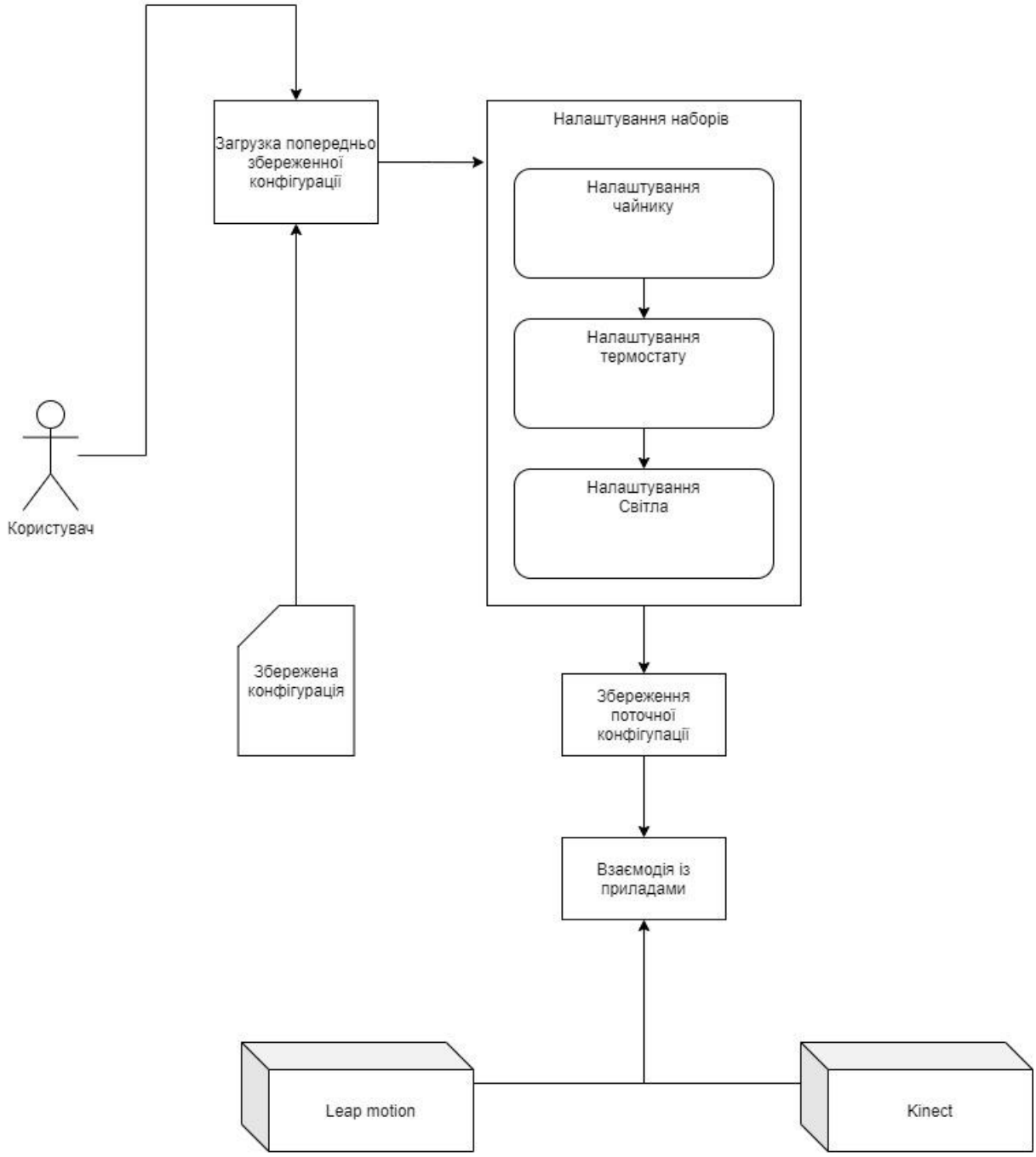


Рис. 1. Діаграма використання системи

3.2. Архітектура системи

Використання моделі Model View Control для персональних програм

Для розробки була обрана мова програмування C#, тому для роботи із *leap motion* та *kinect* були використані бібліотеки *Leap Motion* та *Kinect* [16] для *C#* відповідно.

При розробці графічного інтерфейсу були використані стандартні засоби C# для розробки віконних застосунків – система презентації WPF.

Фрагменти системи виконують такі функції:

- model – зберігає поточний стан обраних жестів і змінює показники приборів. Модель у даному проєкті є основною логічною системою контролю;
- view – графічний інтерфейс користувача, який дозволяю завантажити старий набір жестів, або сконструювати новий для певного користувача із подальшим збереженням;
- control – система контролю приладів за допомогою зчитування жестів із двох використовуваних пристроїв зчитування *kinect*. Для реалізації будуть використані офіційні бібліотеки для розробки застосунків із використанням цих приладів на *C#*.

Для реалізації процесу налаштування алгоритму для відслідковування жестів і контролю приладу був використаний шаблон розробки Стратегія [17].

Інтерфейс стратегії

Був створений інтерфейс стратегії для кожного приладу та для зв’язку засобу зчитування жесту і приладу розумного будинку.

- включення чайнику при реєстрації жесту;
- підвищення і зменшення температури поступово при реєстрації одного з двох жестів відповідно;
- для світла було створено два різних інтерфейса: для контролю світла із двох станів включеного і виключеного, та для контролю як неперервного значення.

Конкретні стратегії

Реалізовані конкретні стратегії являють собою описання алгоритму фіксації жесту і виконують ключову функцію у визначенні способу контролю приладів розумного будинку. Конкретні стратегії є ідентичними для кожного

з приладів, за виключенням використання конкретних алгоритмів, що змінюють стан приладу.

Контекст шаблону

Використовувана конкретна стратегія опису певного жесту визначається контекстом.

Вигляд діаграми шаблону Стратегія можна побачити на рис. 2.

Рис. 2. Загальна діаграма реалізації шаблону Стратегія

Контроль чайнику – це проста операція, що описується лише включенням чайнику для нагрівання води. Тому загальна стратегія управління чайником є найпростішою із розроблених і включає у себе лише функцію активації чайнику.

Опис розроблених методів наведений у табл. 1.

Таблиця 1

Розроблені методи чайнику

Розроблені методи	Описання методів
Abstract gesture	Абстрактний жест, що реалізується у конкретних стратегіях
Turn on kettle	Включає чайник

Контроль температури – операція, що має реєструвати два різних жести і поступово збільшувати або зменшувати температуру. При реєстрації жесту для збільшення і зменшення одночасно, перевага надається жесту для збільшення температури. Тому загальна стратегія має декілька методів контролю.

Опис розроблених методів наведений у табл. 2.

Таблиця 2

Розроблені методи температури

Розроблені методи	Описання методів
Abstract gesture up	Абстрактний жест підвищення температури, що реалізується у конкретних стратегіях
Abstract gesture	Абстрактний жест зниження температури, що реалізується у конкретних стратегіях
Turn up	Збільшує температури, приорітетна функція
Turn down	Знижує температуру

Контроль світла має два варіанта використання: включення або виключення світла повністю або поступово при реєстрації відповідного жесту. Приорітет в обох випадках надається жесту включення світла. Таким чином, загальна стратегія має у собі реалізовані 6 різних методи, що регулюють освітленість

Опис розроблених методів наведений у табл. 3.

Таблиця 3

Розроблені методи освітлення

Розроблені методи	Описання методів
Abstract gesture up	Абстрактний жест підвищення освітленості, що реалізується у конкретних стратегіях
Abstract gesture	Абстрактний жест зниження освітленості, що реалізується у конкретних стратегіях
Turn up	Збільшує освітленість на встановлене значення, приорітетна функція
Turn down	Знижує освітленість на встановлене значення
Turn on	Включає освітленість на максимальне значення, приорітетна функція
Turn off	Повністю виключає освітленість

Зв’язок шаблону Стратегія і схеми MVC [18]

Модель

Головною частиною системи, що зберігає інформацію про обрані методи контролю приладів розумного будинку є модель. Модель також відправляє дані про зміни стану приладів користувачу при зчитуванні відповідних жестів.

Модель містить у собі інформацію про обрану стратегію та всі методи реалізації стратегій.

Інтерфейс користувача

Набори жестів та стан приладів розумного будинку відображається у графічному інтерфейсі користувача, що відображає стан моделі при використанні.

Через інтерфейс користувач обирає конкретний Жест, що буде використовуватися у сесії роботи із приладами розумного будинку у моделі.

Контролер

Прилади зчитування жестів зв’язані із системою через Контролер. У контролері реалізовані алгоритмічно методи відслідковування жестів користувача.

Схему суміжної реалізації двох шаблонів можна побачити на рис. 3.

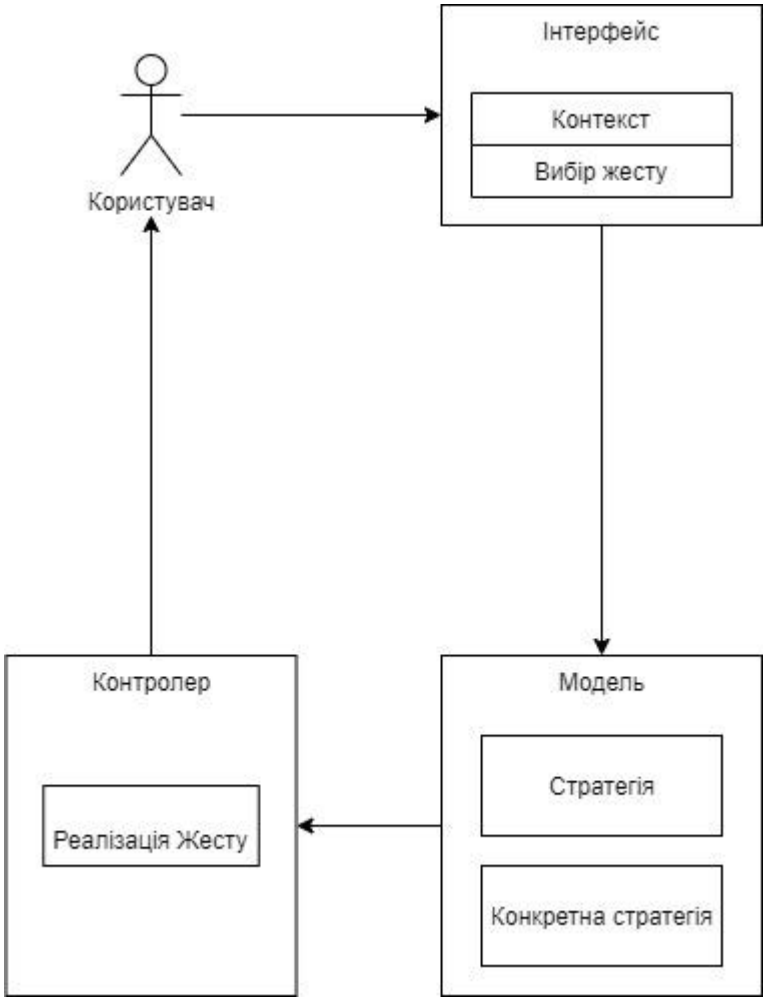


Рис. 3. Схема MVC та Стратегії

3.3. Особливості реалізації

Оскільки розробка даного програмного забезпечення велася за допомогою архітектури MVC, програмне забезпечення можна поділити на відповідні розділи:

- графічний інтерфейс користувача;
- модель контролю схеми жестів;
- забезпечення, що відповідає за зв’язок.

Графічний інтерфейс користувача

При реалізації даного програмного забезпечення було реалізовано два користувацьких вікна. Розробка графічного інтерфейсу була виконана за допомогою системи для побудови користувацьких застосунків WPF і нотації XAML [19].

При запуску застосунку відкривається головна сторінка, на якій виконується побудова і налаштування набору жестів за допомогою зручного меню. Користувач має змогу зберегти набір жестів, так само як і відкрити попередньо сконструйований і збережений набір, відкривши вкладку «File» і натиснувши кнопку «Save» або «Load» відповідно.

Після налаштування набору, виконується запуск системи контролю через натискання кнопки «Start».

Також користувач може відкрити демонстраційне вікно, де він може перевірити працездатність усіх жестів, що наявні у системі, без переналаштування набору за допомогою кнопки «Demo».

Вигляд головного вікна (рис. 4).

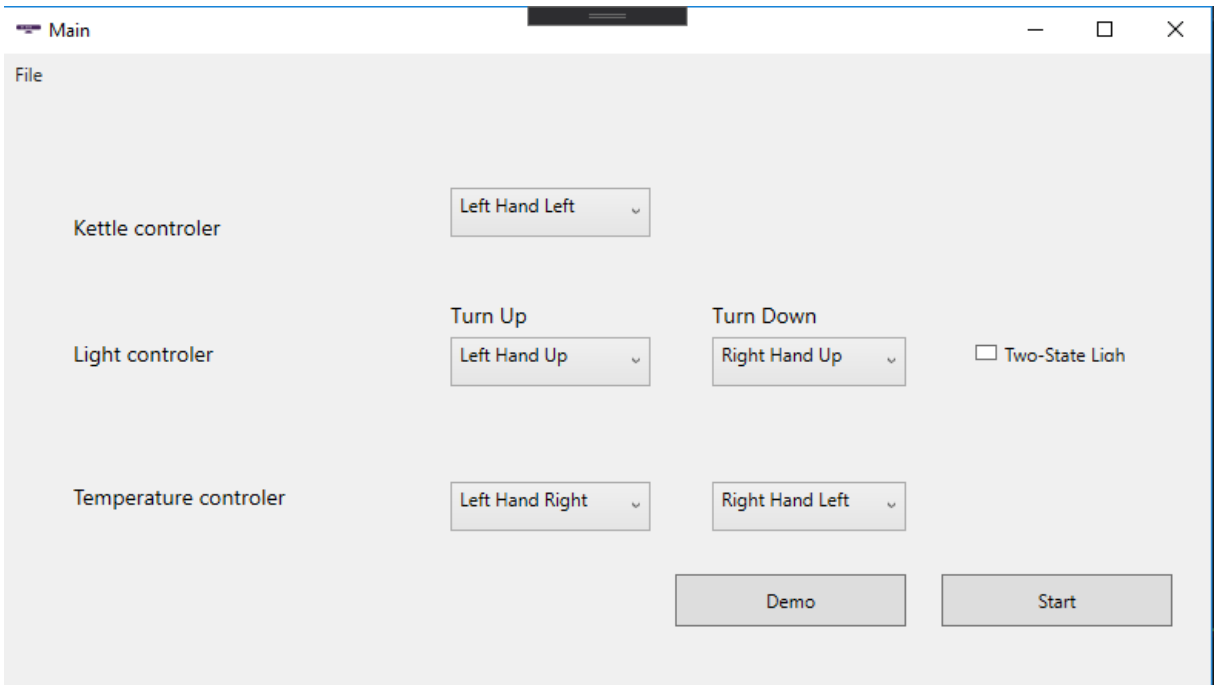


Рис. 4. Головне вікно

Далі відкривається вікно контролю.

Вікно контролю

Вікно контролю відображає віртуальний скелет користувача, що захоплений приладом зчитування Kinect і поведінку та стан приладів, що контролюється за допомогою системи. Таким чином, вікно поділене на дві частини.

У лівій частині вікна відображається скелет, що відслідковується приладом зчитування. Якщо користувач знаходиться у сидячому положенні – він може відключити нижню частину тіла за допомогою поля «Seated mode».

У правій частині вікна відображається стан приладів під час контролю через систему. При зміні стану приладу після зчитування відповідного жесту змінюються і його показники у цій частині вікна. При зміні показника світла, змінюється і колір тексту від чорного при виключеному світлі до жовтого при його повному включені для зручнішого і більш інтуїтивного розуміння поточного стану. Так само, при зміні показників температури, змінюється і колір тексту температури від Синього при найхолоднішій температурі до червоного при найтеплішій обраній температурі.

Вигляд вікна контролю (рис. 5).

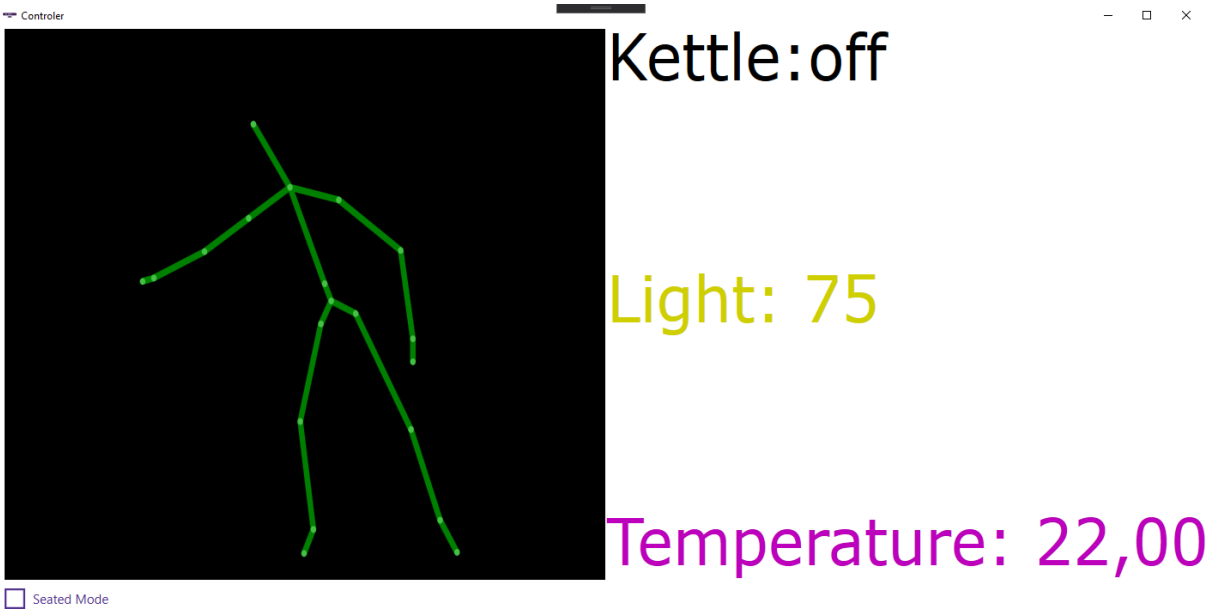


Рис. 5. Вікно контролю

Вікно демонстрації

За структурою вікно демонстрації ідентичне до вікна контролю. Воно так само поділене на дві частини, але права частина відображає лише стан кожного індивідуального жесту і відповідно сповіщає користувача при реєстрації конкретного жесту.

Дане вікно надає можливість протестувати наявні жести і визначити чи виконують вони критерії, наведені нижче. Також, користувач може випробувати жести у умовах перед тим як конструювати свої набори.

Вигляд демонстраційного вікна можна побачити на рис. 6.

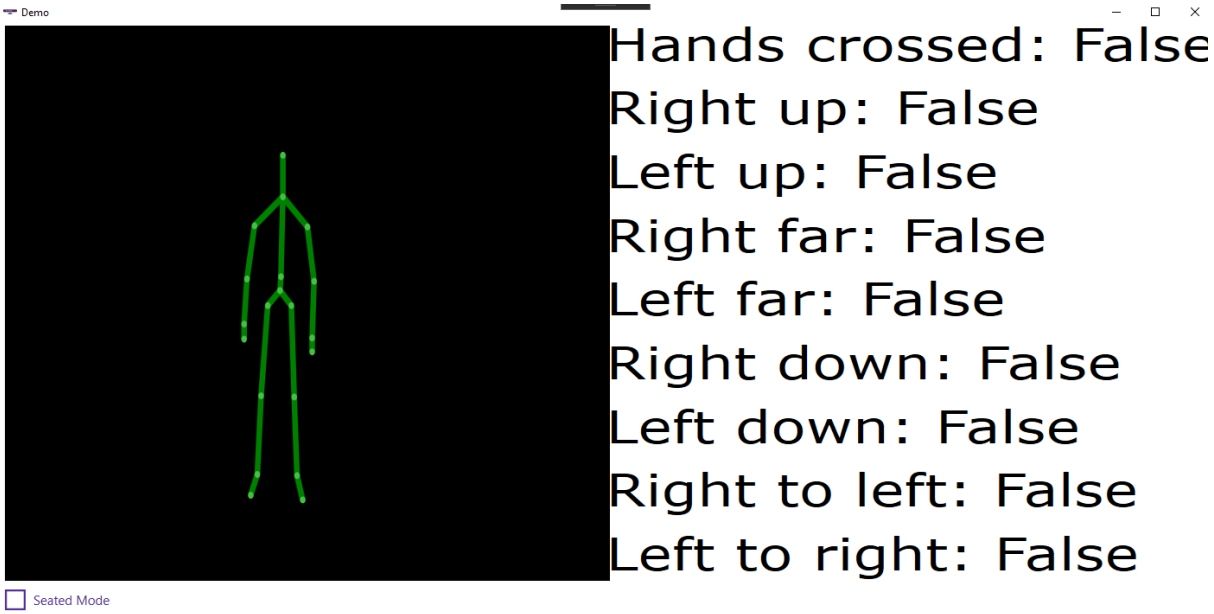


Рис. 6. Демонстраційне вікно

Модель контролю схеми жестів

Основою контролю приладів в даному прокті є відслідковування тіла користувача. Прилад Kinect відслідковує користувача і надає інтерфейс для опрацювання даних із відео потоку.

Обробка даних у *C#* із пристрою Kinect виконується за допомогою підключення, як клас Sensor, що інкапсулює усю інформацію, що отримана із приладу.

При обробці кадру, що отриманий із *Kinect,* виконується пошук скелетів, що пристрій зміг розпізнати внаслідок зчитування з камер. Після виявлення усіх скелетів, будується карта глибини точок, що відповідають точкам кінцівок і з'єднань.

Із карти точок будується з'єднання даних точок для виведення на екран як єдиного об'єкту, після чого скелет відображається на екрані із критичними точками та з'єднаннями між ними.

При врахуванні наведених вище критеріїв, були розроблені 6 унікальних жестів і 5 жестів, що є дзеркальними відображеннями інших. У системі наявні такі жести на вибір користувача:

- занесення лівою або правої руки над головою;

- відведення лівої або правої руки максимально далеко від центру тіла;
- заведення лівої або правої руки за тулуб;
- перехрещування лівої або правої відносно середини плечей таким чином, що рука перетинає центр тіла;
- заведення лівої або правої руки нижче рівня лівого або правого коліна відповідно;
- зведення рук навхрест;

Опис реалізації індивідуальних жестів:

Даний розділ демонструє розроблені жести і показує відсутність колізій між жестами, демонструючи виконання усіх жестів індивідуально.

Занесення руки над головою

Жест реєструється, коли кисть відповідної руки заноситься вище голови по вертикалі. Жест є простим у виконанні і рідко виконується у повсякденному житті. Може бути виконаний людьми із будь-якими фізичними здібностями, тому є достатньо зручним для використання у більшості наборів.

Приклади занесення кісті вище голови можна побачити на рис. 7 і рис.8.

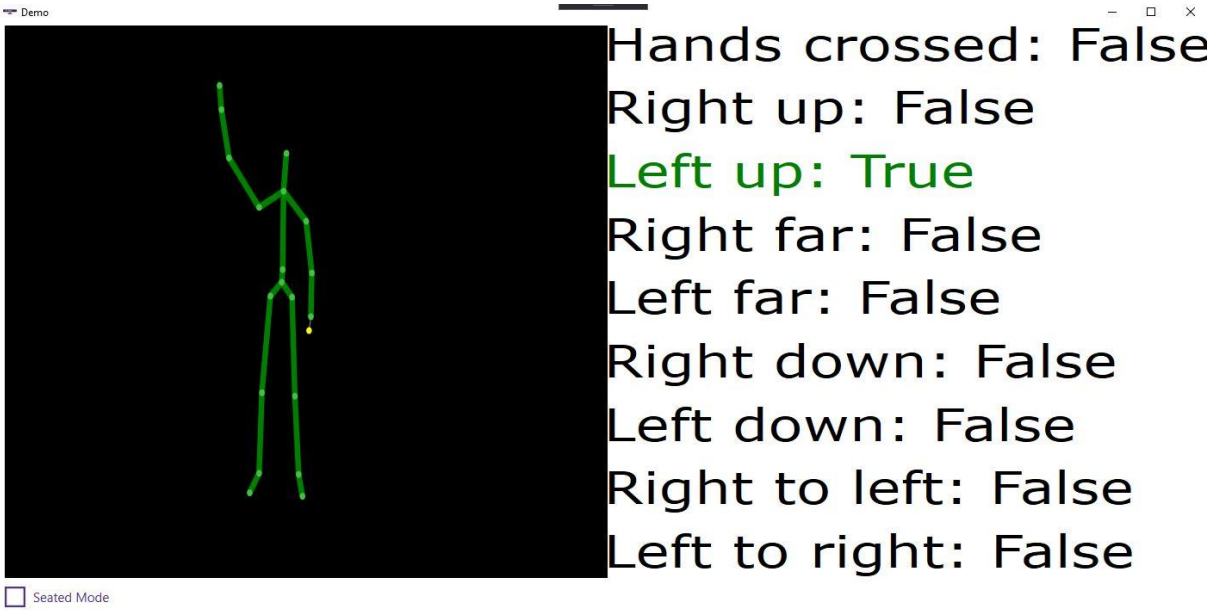


Рис. 7. Занесення правої руки

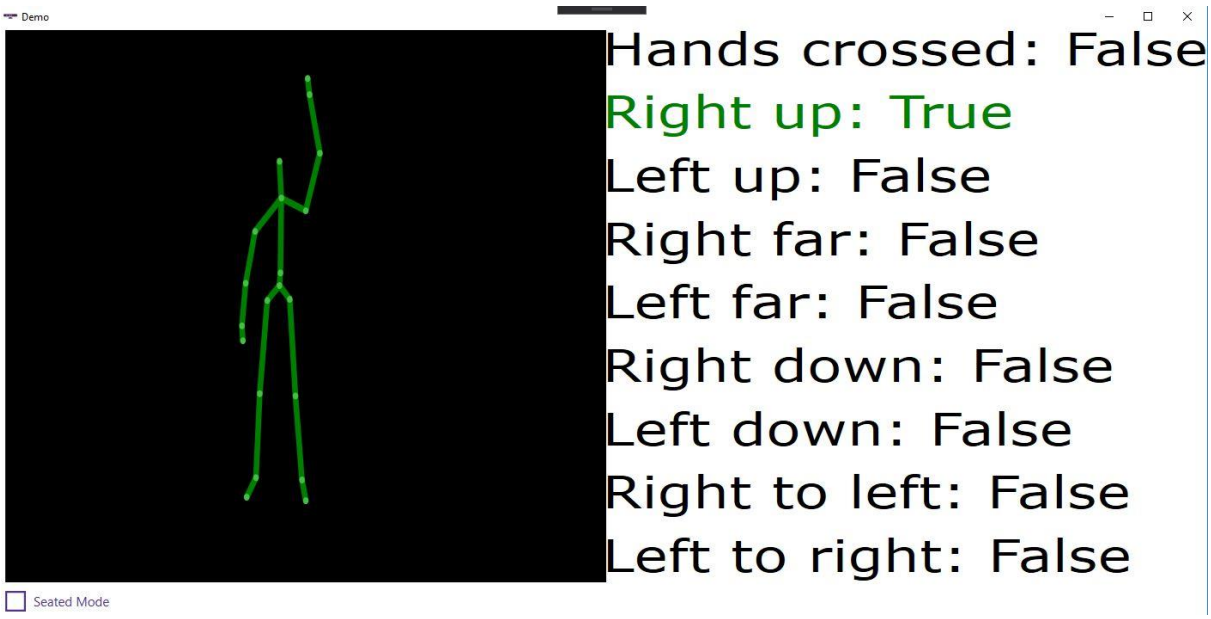


Рис. 8. Занесення правої руки

Відведення руки

Жест реєструється, коли кисть відповідної руки відводиться горизонтально від тіла на достатню відстань. Через те, що відстань вимірюється лише горизонтально, регіон реєстрації даного жесту є вузьким і не буде перетинатися із іншими жестами, що виключає колізії. Жест є простим у виконанні і рідко виконується у повсякденному житті. Може бути виконаний людьми із будь-якими фізичними здібностями, тому є достатньо зручним для використання у більшості наборів.

Приклади відведення кісті від торсу можна побачити на рис. 9 і рис.10.

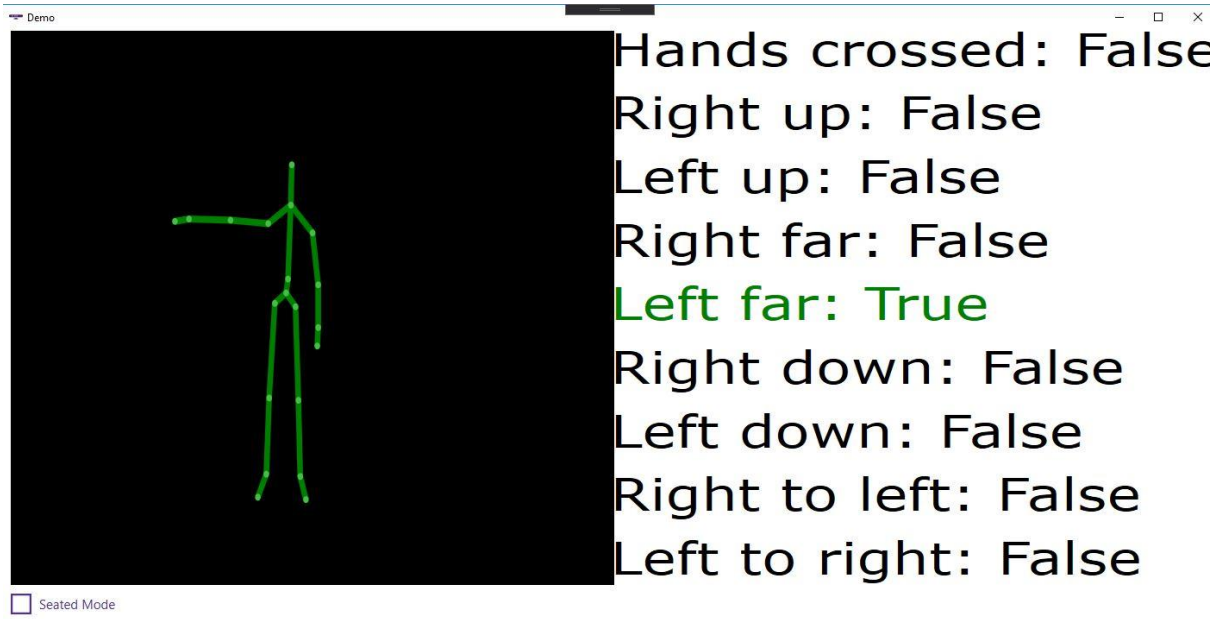


Рис. 9. Відведення лівої руки

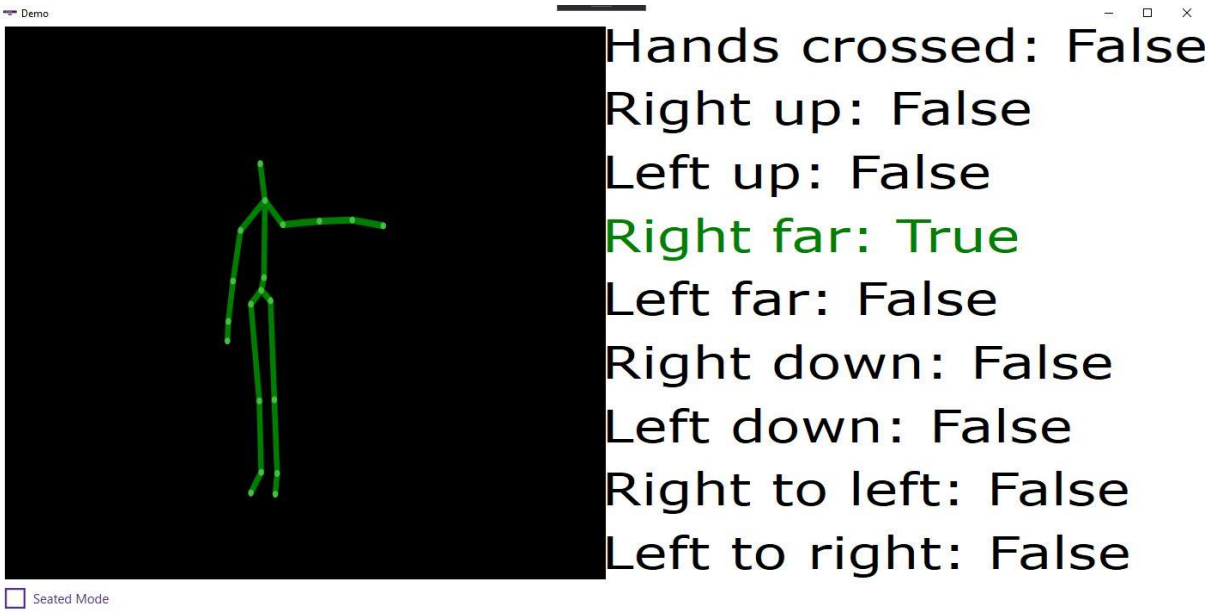


Рис. 10. Відведення правої руки

Заведення руки нижче коліна

Жест реєструється, коли кисть відповідної руки опускається нижче відповідного коліна (правого коліна і руки або лівого коліна і руки). Такий

жест є менш зручним для деяких користувачів, тому він може бути включений не у всі набори. Таке положення тіла рідко відтворюється у повсякденному житті, тому даний жест не виконуватиметься випадково. Положення також є унікальним і не перетинається із іншими жестами і не створює колізій у наборах.

Приклади виконання даного жесту можна побачити на рис. 11.

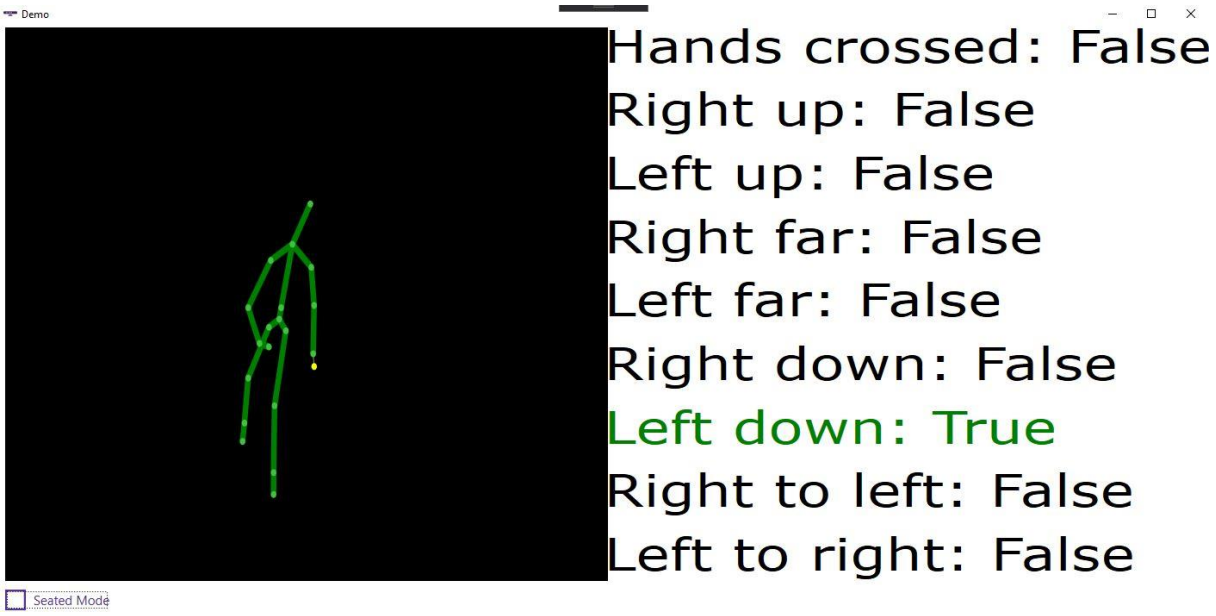


Рис. 11. Заведення лівої руки під коліно

Відповідно, виконання дзеркального жесту можна побачити діла на рис. 12.

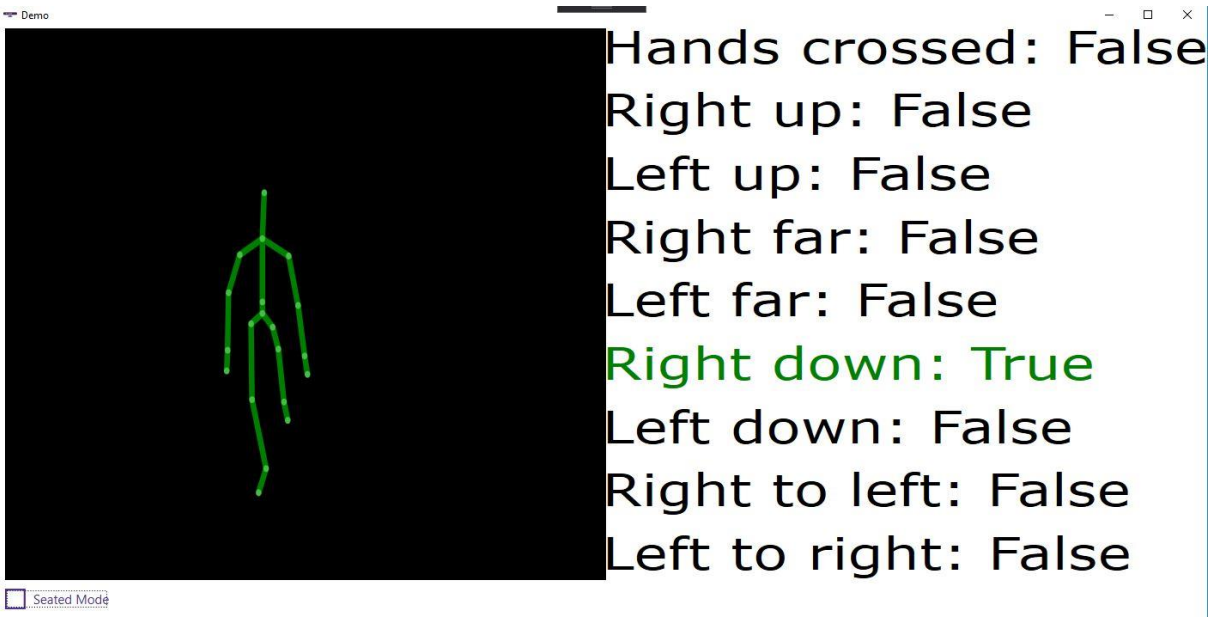


Рис. 12. Заведення правої руки під коліно

Перехрещування лівої або правої із протилежним плечем

Жест реєструється, коли кисть відповідної руки перетинає вертикальне положення протилежного плеча. Оскільки жест реєструється лише при перетині вертикальної лінії плеча, то це виключає колізію із іншими жестами, особливо зхрещуванням рук. Жест є простим у виконанні і рідко виконується у повсякденному житті. Може бути виконаний людьми із будь-якими фізичними здібностями, тому є достатньо зручним для використання у більшості наборів. Виключенням є люди із широкими плечима відносно довжини передпліччя, для яких буде складно заводити руку достатньо далеко для реєстрації жесту.

Приклади виконання даного жесту можна побачити на рис. 13 і рис.14

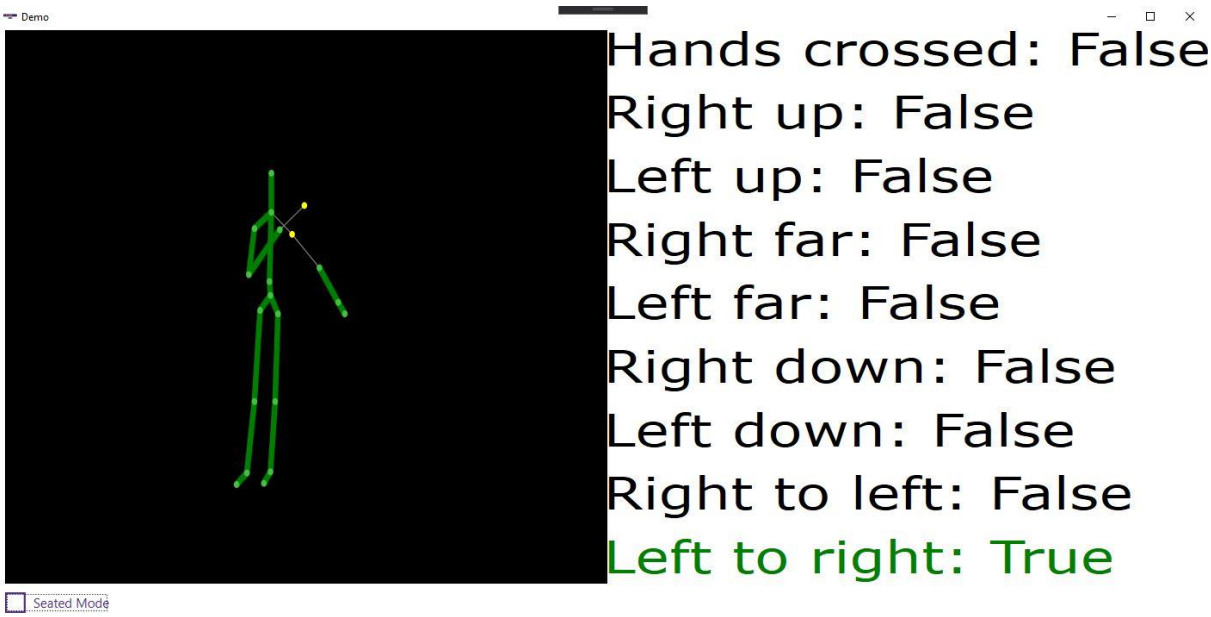


Рис. 13. Заведення лівої руки за праве плече

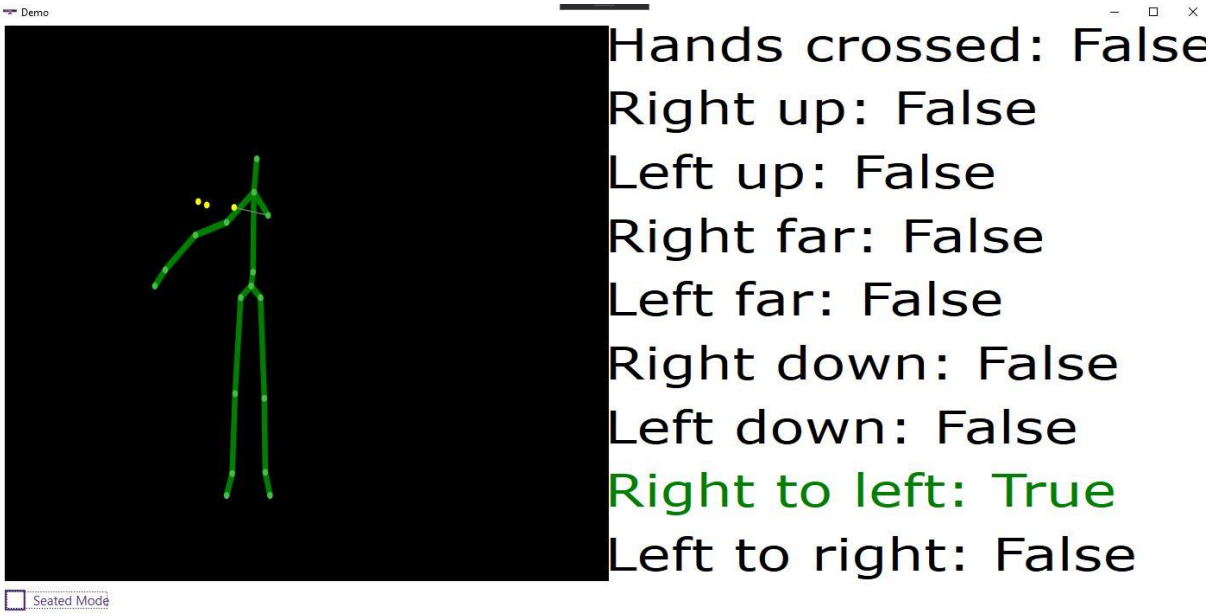


Рис. 14. Заведення правої руки за ліве плече

Зведення рук навхрест

Жест реєструється, коли кисть правої руки знаходиться лівіше кисті лівої руки. Даний жест може бути виконаний у будь-якому регіоні, тому колізії з іншими жестами виникатимуть лише в тому випадку, якщо користувач самостійно вирішить виконати два жести. Жест є простим у виконанні і рідко виконується у повсякденному житті. Може бути виконаний людьми із будь-

якими фізичними здібностями, тому є достатньо зручним для використання у більшості наборів. Виключенням є люди із вузькими плечима, оскільки для таких може виникнути колізія із заведенням кисті за плечі.

Приклади виконання даного жесту можна побачити на рис. 15.

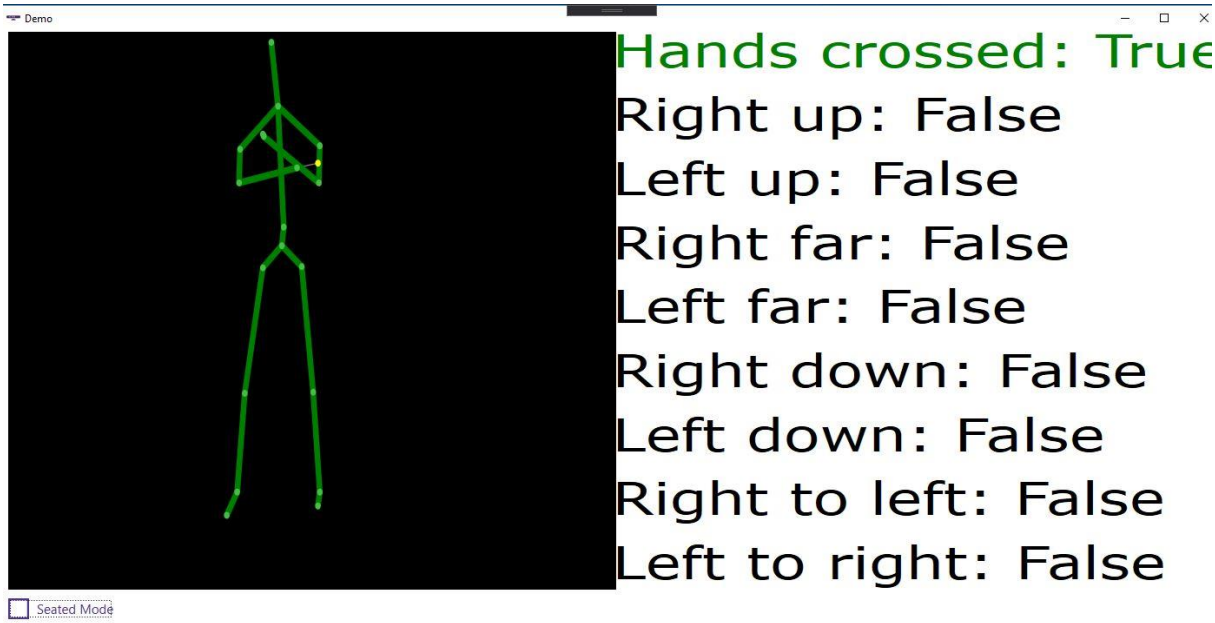


Рис. 15. Зведення рук навхрест

Заведення руки за тулуб

Жест реєструється, коли кисть відповідної руки заводиться за тулуб таким чином, що відповідна кисть перестає реєструватися. Оскільки інші жести, представлені у даному проєкті, реєструються лише при відслідковуванні кінцівки, даний жест не викликає колізій. Жест є простим у виконанні і рідко виконується у повсякденному житті. Може бути виконаний людьми із будь-якими фізичними здібностями, тому є достатньо зручним для використання. Даний жест є найбільш проблематичним, оскільки він може бути виконаний при великій кількості інших можливих взаємодій, тому він, вірогідно, не буде включатися користувачами у набори. Він представлений у даному проєкті як приклад вузьконаправленого жесту, який буде використаний невеликою групою користувачів.

Приклади виконання даного жесту можна побачити на рис. 16 і рис.17.

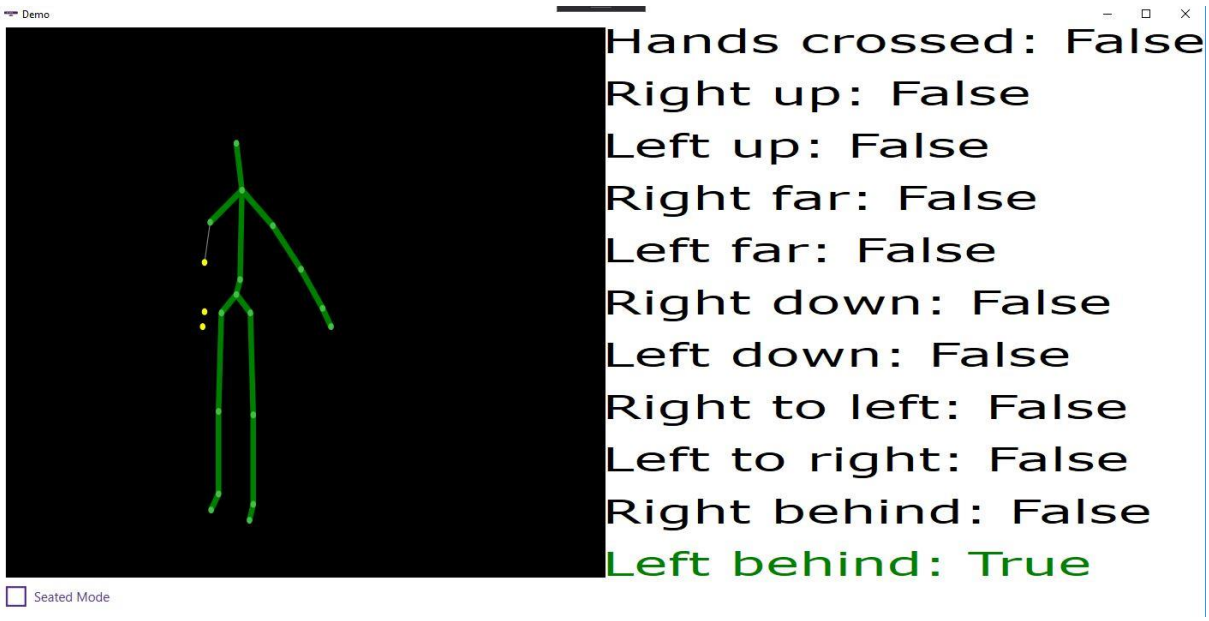


Рис. 16. Заведення лівої руки за тіло

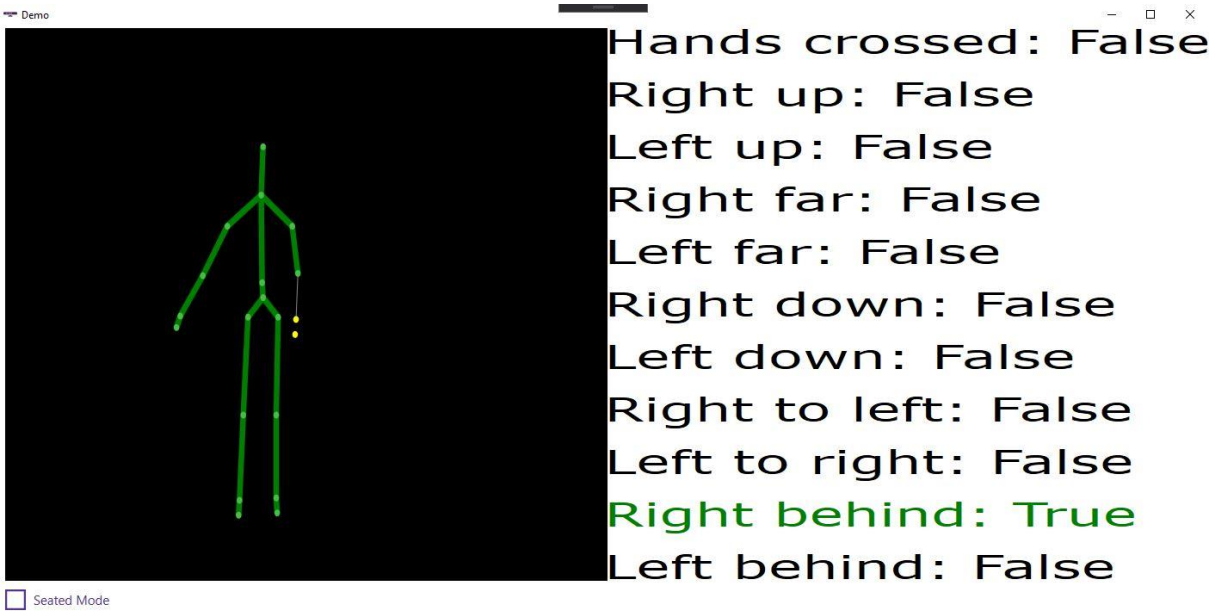


Рис. 17. Заведення правої руки за тіло

3.4. Висновки до розділу

У даному розділі було проведено загальний опис розроблюваного програмного забезпечення у вигляді десктопного додатку. Також були описані використані класи для реєстрації жестів.

Був описаний формат файлів, що зберігає дані про сконструйовані користувачем набори жестів.

Були описані способи реєстрації кожного індивідуального жесту і проведений аналіз жестів, згідно із необхідними вимогами до розроблюваних жестів для нормального функціонування системи.

При описі архітектури системи було проведено аналогію із загальновживаною моделлю MVC (“Model-View-Controller”).

4. АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ КОНТРОЛЮ

4.1. Аналіз реалізованої системи

У результаті розробки системи управління приладами розумного будинку за допомогою жестів було створено програмне забезпечення, призначене для використання вдома разом із приладами зчитування жестів для контролю побутових приладів.

Відповідно до сформованих функціональних вимог до системи (див. підрозділ 1.4. “Загальні вимоги до системи”) було розроблено відповідні рішення:

- реалізовано модель MVC (Model View Control) для розмежування функціональні можливості системи на різні модулі: відображення результату контролю на View сегменті, зміна даних в результаті контролю на сегменті Model та контроль приладів на сегменті Control;
- при розробці даного дипломного проєкту було реалізовано функціональні можливості для перегляду результату контролю приладів;
- реалізовано функціональні можливості контролю приладів за допомогою приладів зчитування жестів;
- у даному ПЗ була створена сторінка для налаштування різних наборів жестів і різних типів взаємодії із наявними приладами розумного будинку;
- при розробці даного ПЗ було створено графічний інтерфейс для відтворення і зручного представлення результатів контролю приладів розумного будинку;
- ПЗ включає у себе можливість обирати різні зібрані набори, зберігати та відтворювати збережені набори для підвищення зручності користування декількома користувачами системи на різних пристроях.

4.2. Тестування системи контролю

Тестування розроблюваного програмного забезпечення проводилося на кожній ітерації створення системи та є невід'ємною частиною процесу створення закінченого програмного продукту. В ході виконання дипломного проєкту та розроблення системи контролю воно виконувалось після кожного етапу реалізації певних елементів системи.

Тестування розроблюваного програмного забезпечення проводилося на кожній ітерації створення системи та є невід'ємною частиною процесу створення закінченого програмного продукту. В ході виконання дипломного проєкту та розроблення десктопного застосунку воно виконувалось після кожного етапу реалізації певних елементів системи.

Головним способом тестування було прийнято ручне тестування для перевірки працездатності функціональні можливості і можливості зчитування з людей різного росту або комплекції.

4.3. Порівняння розробки із існуючими аналогами

У першому розділі дипломного проєкту було розглянуто декілька аналогів, що використовуються для взаємодії із розумним будинком. Альтернативні варіанти взаємодії були вивчені з метою виявлення альтернативних способів контролю, що не дозволяють використовувати їх певному сегменту користувачів. Таким чином, дана система може бути використана як окрема єдина система, або інтегровна у існуючу систему контролю для розширення методів взаємодії із побутовими приладами.

Основні способи:

- голосове управління – найпопулярніший спосіб управління для сучасних розумних будинків, що дозволяє зручно використовувати як прилади, так і онлайн-системи особистого органайзеру. Недоліками даної системи є неможливість управління людьми з

вадами мовлення і потребує локалізації, що сповільнює використання даної технології;

- управління за допомогою датчиків – такий спосіб управління є достатньо популярним і використовується у поєднанні із іншими способами контролю приладів. Даний спосіб є найстарішим, оскільки він використовувався до розробки технології розумного будинку і включав у себе навіть механічні аналоги. Недоліком цього способу є низька адаптивність і складна інтеграція у систему розумного будинку, що не дозволяє контролювати широкий спектр побутових приладів використовуючи лише датчики управління;
- Контроль через вбудовані інтерфейси – перший спосіб, що був використаний як частина розумного будинку. Управління виконується через стандартні перемикачі, такі як пульт або звичайні кнопки і тумблера.

Таким чином було прийняте рішення розробити адаптивну систему управління, що дозволить використання побутових приладів незалежно від регіону використання і надаючи більше можливостей для тих користувачів, що не можуть використовувати голосовий контроль.

4.4. Рекомендації щодо подальшого вдосконалення

Дана система контролю має широкі можливості розширення для покращення зручності використання і інтеграції більшої кількості приладів та способів контролю.

Основні можливості розширення функціональних можливостей:

- збільшення кількості побутових приладів, що можуть бути контрольовані за допомогою існуючих рішень для приладів, що вже використовувані в системі;
- розробка нових жестів контролю для розширення адаптивності до вимог користувача;

- введення додаткових приладів контролю і розробка нових способів контролю;
- інтеграція розробленої системи у існуючі системи контролю розумного будинку.

Таким чином, розроблена система має широкі можливості для розширення функціональних можливостей і покращення взаємодії.

4.5. Висновки до розділу

У даному розділі було проведено аналіз реалізованого програмного забезпечення для контролю приладів розумного будинку за допомогою жестів. Було проведено аналогію із поставленими функціональними і нефункціональними вимогами та відповідно описано методи та рішення, що були прийняті, і яким способом вони розроблені.

На кожній ітерації створення системи проводилося ручне тестування та вносилися відповідні правки. Наприкінці було проведено тестування реальними потенційними користувачами. Було враховано усі недоліки, що виникли, та побажання користувачів щодо функціональних можливостей та зручності графічного інтерфейсу і реалізовано їх у розроблюваній системі.

ВИСНОВКИ

Метою даного дипломного проєкту було створення програмної системи, що забезпечує контроль приладів за технологією Smart Home через прилади зчитування жестів. Це забезпечить користувачам більш зручну взаємодію із своїм будинком і розширить аудиторію, що зможе користуватися даною технологією.

Був проведений аналіз існуючих способів взаємодії із приладами, плюси та недоліки представлених рішень, виявлено потенційних користувачів, для яких буде більш зручна розроблювана система із даного проєкту.

Важливим пунктом при аналізі технологій і приладів, що були використані у даному проєкті, був пошук зручних і розповсюджених приладів, що уже використовуються великим колом користувачів, для спрощення інтеграції розробленого ПЗ.

Після порівняння засобів реалізації, було сформовано вимоги для побудови десктопного програмного забезпечення, написанного мовою програмування *C#* із використанням стандартних засобів створення графічного інтерфейсу користувача Windows Presentation Foundation та бібліотек для розробки програмного забезпечення із використанням приладів зчитування жестів *Kinect*.

Було створене програмне забезпечення для використання із технологією Smart Home для домашнього використання. Розроблений десктопний застосунок відповідає наступним вимогам:

- система контролює три різних прилади: Чайник, Світло та Термостат;
- користувач може налаштовувати за його бажанням набір жестів, що буде відповідати за контроль кожного приладу;
- збереження налаштувань у зручному форматі для повторного використання і перенесення попередньо побудованих користувачем наборів жестів.

Розробка програмного забезпечення виконана у повному обсязі та відповідає поставленим вимогам. Тестування системи виконано у відповідності до затвердженої програми та методики тестування.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

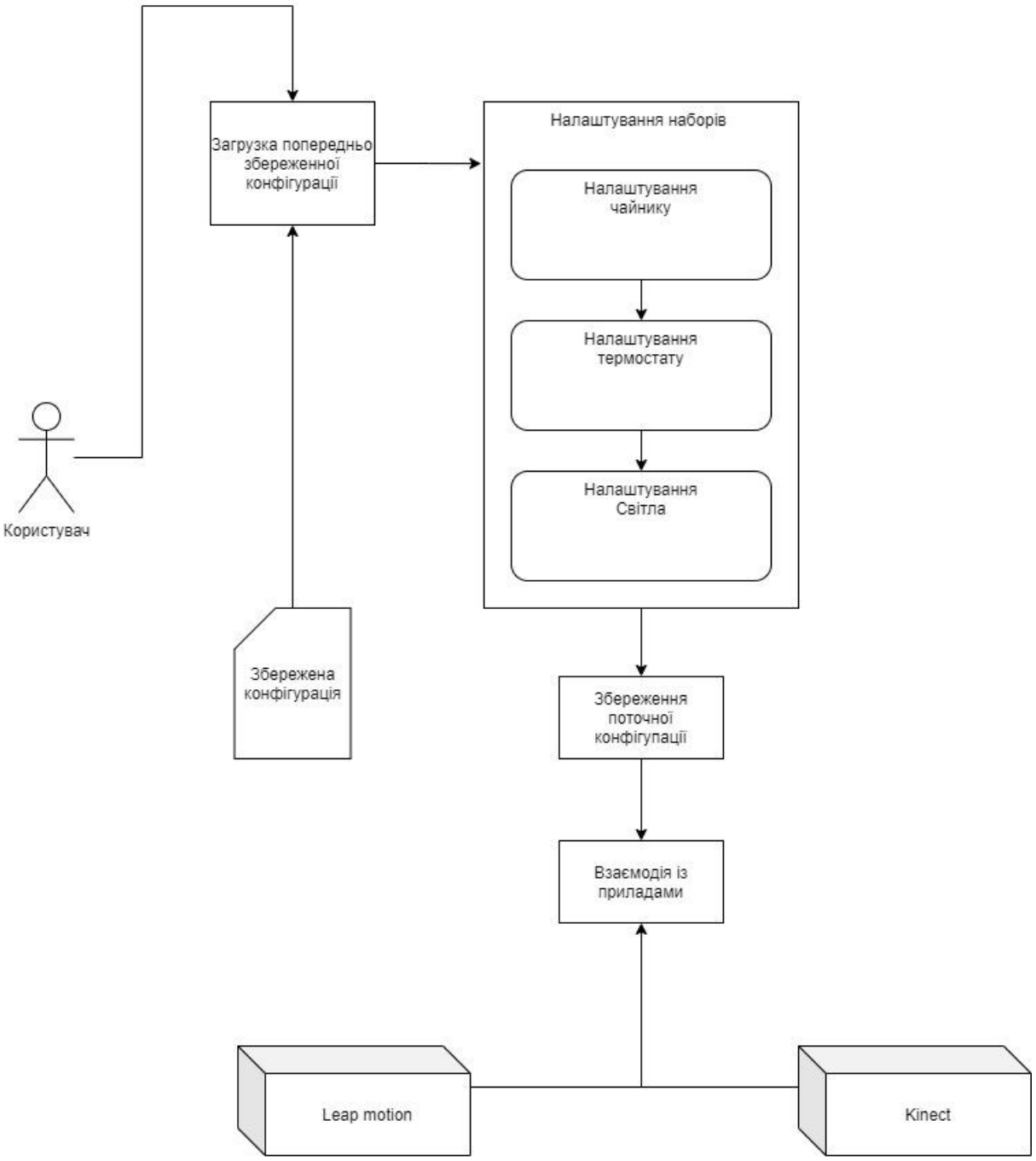
1. Kinect documentation : [Електронний ресурс]. Режим доступу: https://developer.microsoft.com/en-us/windows/kinect/
2. Leap Motion documentation : [Електронний ресурс]. Режим доступу: https://developer.leapmotion.com/
3. AJAX : [Електронний ресурс]. Режим доступу: https://english.ajax.nl/streams/ajax-now.htm
4. JSON : [Електронний ресурс]. Режим доступу: https://www.json.org/json-en.html
5. XML documentation : [Електронний ресурс]. Режим доступу: https://www.xml.com/
6. SGML Resources: [Електронний ресурс]. Режим доступу: https://www.w3.org/MarkUp/SGML/Overview.html
7. MVC in C# : [Електронний ресурс]. Режим доступу: https://dotnet.microsoft.com/apps/aspnet/mvc
8. WPF documentation: [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/en-us/dotnet/framework/wpf/
9. XAML in C# [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/en-us/dotnet/desktop-wpf/fundamentals/xaml
10. American Housebuilder Association [Електронний ресурс]. Режим доступу: https://www.nahb.org/
11. Java [Електронний ресурс]. Режим доступу: https://www.java.com/
12. Smart Home [Електронний ресурс]. Режим доступу: https://www.smarthome.com/
13. C# [Електронний ресурс]. Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp/
14. C++ [Електронний ресурс]. Режим доступу: http://cplusplus.com/
15. Python [Електронний ресурс]. Режим доступу: https://www.python.org/

16. Kinect SDK [Електронний ресурс]. Режим доступу: https://www.microsoft.com/en-us/download/details.aspx?id=40278
17. Robert C. Martin. Clean Architecture: A Craftsman's Guide to Software Structure and Design [Текст]. / Robert C. Martin – 1st edition. – Prentice Hall, 2017 – 428 p.
18. Erich Gamma, Richard Helm , Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software [Текст]. / Erich Gamma, Richard Helm , Ralph Johnson, John Vlissides – 1st edition. – Addison-Wesley Professional, 1994 – 416 p.
19. Dimitri Laslo. Developing multi-platform desktop applications with .NET Core 3 and Visual Studio 2019. [Текст] / Laslo Dimitri – Independently published, 2019 – 321 p.

ДОДАТКИ

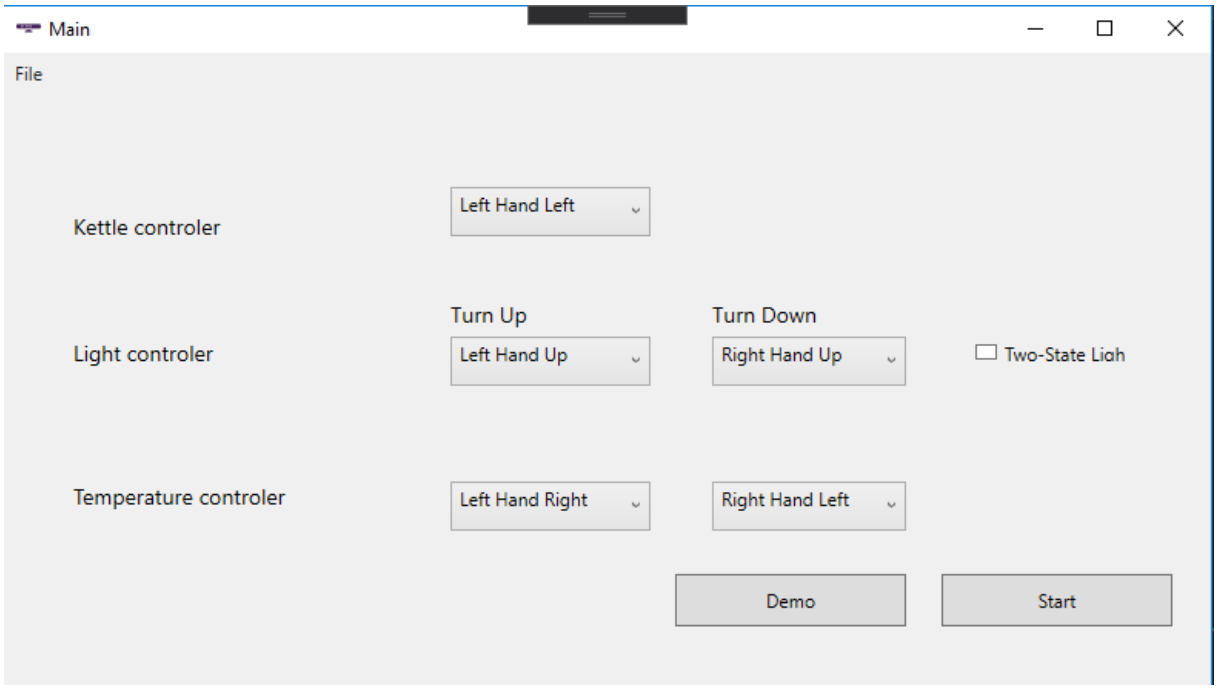
Додаток 1

Копії графічних матеріалів

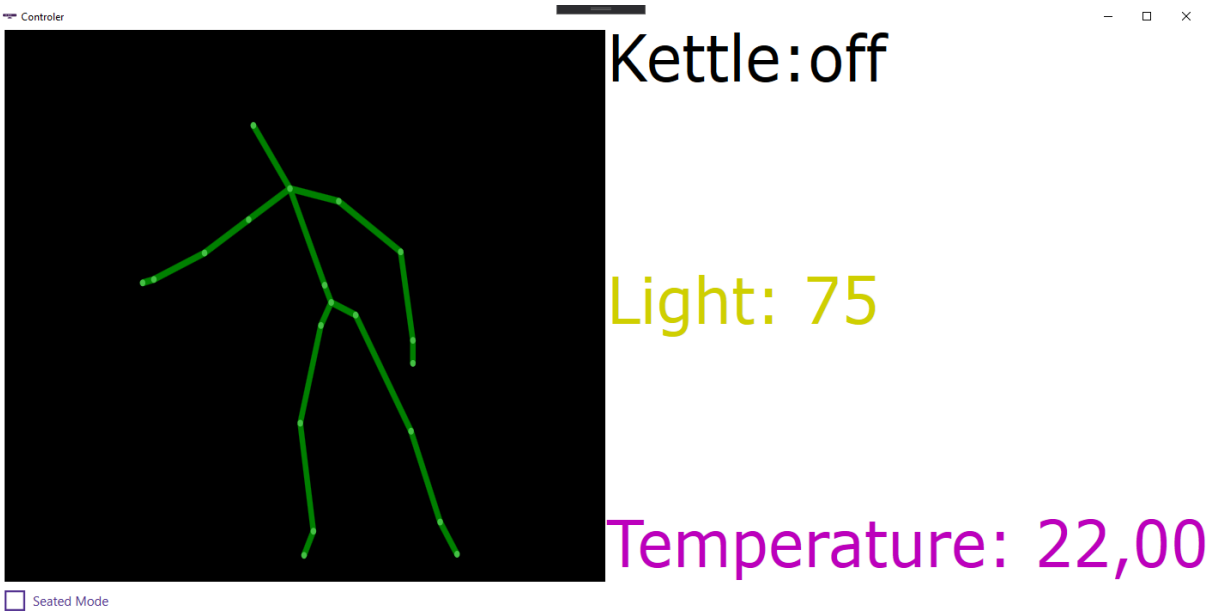


ДП.045440-06-99
Програмна система Адаптивного управління Пристроями за технологією Smart Home. Загальна архітектура системи. Діаграма використання системи.

ДП.045440-07-99
Програмна система
Адаптивного управління
Пристроями за технологією
Smart Home. Загальна
архітектура системи.
Розроблена система.



Коломієць Денис, група КП-61



Коломієць Денис, група КП-61

Додаток 2

Лістинг програм

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.Remoting.Contexts;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace Microsoft.Samples.Kinect.SkeletonBasics
{
    /// <summary>
    /// Логика взаимодействия для Window1.xaml
    /// </summary>
    public partial class Window1 : Window
    {

        ComboBox combo1;
        ComboBox combo2;
        ComboBox combo3;
        ComboBox combo4;
        ComboBox combo5;


        public Window1()
        {
            InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            Context s1 = new Strategy();
            Context s2 = new Strategy();
            Context s3 = new Strategy();
            Context s4 = new Strategy();
            Context s5 = new Strategy();

            s1.setStrategy(combo1.Text);
```

```
            s2.setStrategy(combo2.Text);
            s3.setStrategy(combo3.Text);
            s4.setStrategy(combo4.Text);
            s5.setStrategy(combo5.Text);

            List<Context> contexts = new List<Context> { s1, s2, s3, s4, s5
};

            MainWindow win2 = new MainWindow(contexts);
            win2.Show();
        }

        private void Demo_Button_Click(object sender, RoutedEventArgs e)
        {

            DemoWindow win3 = new DemoWindow();
            win3.Show();
        }
    }
}
namespace Microsoft.Samples.Kinect.SkeletonBasics
{
    public interface IStrategy
    {
        void Use();
    }

    public class Context
    {
        public interface IStrategy
        {
            void Algorithm();
        }

        public class ConcreteStrategy1 : IStrategy
        {
            public void Algorithm()
            {
                kattle = true;
            }
        }

        public class ConcreteStrategy2 : IStrategy
```

```
{
    public void Algorithm()
    {
        light += 4;
    }
}

public class ConcreteStrategy3 : IStrategy
{
    public void Algorithm()
    {
        light -= 4;
    }
}

public class ConcreteStrategy4 : IStrategy
{
    public void Algorithm()
    {
        temperature += 4;
    }
}

public class ConcreteStrategy5 : IStrategy
{
    public void Algorithm()
    {
        temperature -= 4;
    }
}

public class Context
{
    private IStrategy _strategy;

    public Context(IStrategy strategy)
    {
        _strategy = strategy;
    }

    public void SetStrategy(IStrategy strategy)
    {
        _strategy = strategy;
```

```
            }

            public void ExecuteOperation()
            {
                _strategy.Algorithm();
            }
        }
}

//------------------------------------------------------------------------------
---
// <copyright file="MainWindow.xaml.cs" company="Microsoft">
//     Copyright (c) Microsoft Corporation.  All rights reserved.
// </copyright>
//------------------------------------------------------------------------------
---

namespace Microsoft.Samples.Kinect.SkeletonBasics
{
    using System.IO;
    using System.Windows;
    using System.Windows.Media;
    using Microsoft.Kinect;
    using System;
    using System.Globalization;

    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        /// <summary>
        /// Width of output drawing
        /// </summary>
        private const float RenderWidth = 640.0f;

        /// <summary>
        /// Height of our output drawing
        /// </summary>
        private const float RenderHeight = 480.0f;

        /// <summary>
        /// Thickness of drawn joint lines
```

```
        /// </summary>
        private const double JointThickness = 3;

        /// <summary>
        /// Thickness of body center ellipse
        /// </summary>
        private const double BodyCenterThickness = 10;

        /// <summary>
        /// Thickness of clip edge rectangles
        /// </summary>
        private const double ClipBoundsThickness = 10;

        private bool kattle = false;

        private int lightVal = 750;

        private int tempVal = 500;

        /// <summary>
        /// Brush used to draw skeleton center point
        /// </summary>
        private readonly Brush centerPointBrush = Brushes.Blue;

        /// <summary>
        /// Brush used for drawing joints that are currently tracked
        /// </summary>
        private readonly Brush trackedJointBrush = new
SolidColorBrush(Color.FromArgb(255, 68, 192, 68));

        /// <summary>
        /// Brush used for drawing joints that are currently inferred
        /// </summary>
        private readonly Brush inferredJointBrush = Brushes.Yellow;

        /// <summary>
        /// Pen used for drawing bones that are currently tracked
        /// </summary>
        private readonly Pen trackedBonePen = new Pen(Brushes.Green, 6);

        /// <summary>
        /// Pen used for drawing bones that are currently inferred
        /// </summary>
```

```
        private readonly Pen inferredBonePen = new Pen(Brushes.Gray, 1);

        /// <summary>
        /// Active Kinect sensor
        /// </summary>
        private KinectSensor sensor;

        /// <summary>
        /// Drawing group for skeleton rendering output
        /// </summary>
        private DrawingGroup drawingGroup;

        private DrawingGroup drawingGroup2;

        /// <summary>
        /// Drawing image that we will display
        /// </summary>
        private DrawingImage imageSource;

        private DrawingImage kettleSource;


        /// <summary>
        /// Initializes a new instance of the MainWindow class.
        /// </summary>
        public MainWindow()
        {
            InitializeComponent();

        }

        /// <summary>
        /// Draws indicators to show which edges are clipping skeleton data
        /// </summary>
        /// <param name="skeleton">skeleton to draw clipping information
for</param>
        /// <param name="drawingContext">drawing context to draw to</param>
        private static void RenderClippedEdges(Skeleton skeleton,
DrawingContext drawingContext)
        {
            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Bottom))
            {
                drawingContext.DrawRectangle(
```

```
                    Brushes.Red,
                    null,
                    new Rect(0, RenderHeight - ClipBoundsThickness,
RenderWidth, ClipBoundsThickness));
            }

            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Top))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(0, 0, RenderWidth, ClipBoundsThickness));
            }

            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Left))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(0, 0, ClipBoundsThickness, RenderHeight));
            }

            if (skeleton.ClippedEdges.HasFlag(FrameEdges.Right))
            {
                drawingContext.DrawRectangle(
                    Brushes.Red,
                    null,
                    new Rect(RenderWidth - ClipBoundsThickness, 0,
ClipBoundsThickness, RenderHeight));
            }
        }

        /// <summary>
        /// Execute startup tasks
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void WindowLoaded(object sender, RoutedEventArgs e)
        {
            // Create the drawing group we'll use for drawing
            this.drawingGroup = new DrawingGroup();

            this.drawingGroup2 = new DrawingGroup();
```

```
            // Create an image source that we can use in our image control
            this.imageSource = new DrawingImage(this.drawingGroup);

            this.kettleSource = new DrawingImage(this.drawingGroup2);
            // Display the drawing using our image control
            Image.Source = this.imageSource;

            KettleImage.Source = this.kettleSource;

            // Look through all sensors and start the first connected one.
            // This requires that a Kinect is connected at the time of app
startup.
            // To make your app robust against plug/unplug,
            // it is recommended to use KinectSensorChooser provided in
Microsoft.Kinect.Toolkit (See components in Toolkit Browser).
            foreach (var potentialSensor in KinectSensor.KinectSensors)
            {
                if (potentialSensor.Status == KinectStatus.Connected)
                {
                    this.sensor = potentialSensor;
                    break;
                }
            }

            if (null != this.sensor)
            {
                // Turn on the skeleton stream to receive skeleton frames
                this.sensor.SkeletonStream.Enable();

                // Add an event handler to be called whenever there is new
color frame data
                this.sensor.SkeletonFrameReady +=
this.SensorSkeletonFrameReady;

                // Start the sensor!
                try
                {
                    this.sensor.Start();
                }
                catch (IOException)
                {
                    this.sensor = null;
```

```
                }
            }
        }

        /// <summary>
        /// Execute shutdown tasks
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void WindowClosing(object sender,
System.ComponentModel.CancelEventArgs e)
        {
            if (null != this.sensor)
            {
                this.sensor.Stop();
            }
        }

        /// <summary>
        /// Event handler for Kinect sensor's SkeletonFrameReady event
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
        private void SensorSkeletonFrameReady(object sender,
SkeletonFrameReadyEventArgs e)
        {
            Skeleton[] skeletons = new Skeleton[0];

            using (SkeletonFrame skeletonFrame = e.OpenSkeletonFrame())
            {
                if (skeletonFrame != null)
                {
                    skeletons = new
Skeleton[skeletonFrame.SkeletonArrayLength];
                    skeletonFrame.CopySkeletonDataTo(skeletons);
                }
            }

            using (DrawingContext dc = this.drawingGroup.Open())
            {
                // Draw a transparent background to set the render size
                dc.DrawRectangle(Brushes.Black, null, new Rect(0.0, 0.0,
RenderWidth, RenderHeight));
```

```
                if (skeletons.Length != 0)
                {
                    foreach (Skeleton skel in skeletons)
                    {
                        RenderClippedEdges(skel, dc);

                        if (skel.TrackingState ==
SkeletonTrackingState.Tracked)
                        {
                            this.DrawBonesAndJoints(skel, dc);
                        }
                        else if (skel.TrackingState ==
SkeletonTrackingState.PositionOnly)
                        {
                            dc.DrawEllipse(
                            this.centerPointBrush,
                            null,
                            this.SkeletonPointToScreen(skel.Position),
                            BodyCenterThickness,
                            BodyCenterThickness);
                        }
                    }
                }

                // prevent drawing outside of our render area
                this.drawingGroup.ClipGeometry = new RectangleGeometry(new
Rect(0.0, 0.0, RenderWidth, RenderHeight));
            }

            using (DrawingContext dc = this.drawingGroup2.Open())
            {
                // Draw a transparent background to set the render size

                int light = (int)lightVal / 10;

                double temp = tempVal / 83.3 + 16;

                String k;
                if (kattle) {
                    k = "On";
                } else
                {
```

```
                k = "off";
            }

            Point p1 = new Point(0, 100);
            Point p2 = new Point(0, 250);
            Point p3 = new Point(0, 400);

            Color c1 = Color.FromScRgb(1, (float)lightVal / 1200,
(float)lightVal / 1200, 0);
            Color c2 = Color.FromScRgb(1, (float)tempVal/1000, 0,
(float)(1000-tempVal)/1000);

            FormattedText formattedText = new FormattedText(
                "Kettle:" + k,
                CultureInfo.GetCultureInfo("en-us"),
                FlowDirection.LeftToRight,
                new Typeface("Verdana"),
                40,
                Brushes.Black);
            FormattedText formattedText2 = new FormattedText(
                "Light: " + light.ToString(),
                CultureInfo.GetCultureInfo("en-us"),
                FlowDirection.LeftToRight,
                new Typeface("Verdana"),
                40,
                new SolidColorBrush(c1));
            FormattedText formattedText3 = new FormattedText(
                "Temperature: " + temp.ToString("N"),
                CultureInfo.GetCultureInfo("en-us"),
                FlowDirection.LeftToRight,
                new Typeface("Verdana"),
                40,
                new SolidColorBrush(c2));

            dc.DrawText(formattedText, p1);
            dc.DrawText(formattedText2, p2);
            dc.DrawText(formattedText3, p3);
            if (skeletons.Length != 0)
            {
                foreach (Skeleton skel in skeletons)
                {
```

```
                        if (skel.TrackingState ==
SkeletonTrackingState.Tracked)
                        {
                            if (skel.Joints[JointType.Head].Position.Y <
skel.Joints[JointType.HandRight].Position.Y)
                            {
                                if(lightVal < 1000)
                                {
                                    lightVal += 4;
                                }
                            }  else if
(skel.Joints[JointType.Head].Position.Y <
skel.Joints[JointType.HandLeft].Position.Y)
                            {
                                if (lightVal > 0)
                                {
                                    lightVal -= 4;
                                }
                            }

                            if
(skel.Joints[JointType.ShoulderCenter].Position.X >
skel.Joints[JointType.HandRight].Position.X)
                            {
                                if (tempVal < 1000)
                                {
                                    tempVal += 4;
                                }
                            }
                            else if
(skel.Joints[JointType.ShoulderCenter].Position.X <
skel.Joints[JointType.HandLeft].Position.X)
                            {
                                if (tempVal > 0)
                                {
                                    tempVal -= 4;
                                }
                            }
                            if
(Math.Abs(skel.Joints[JointType.ShoulderCenter].Position.X -
skel.Joints[JointType.HandRight].Position.X) > 0.65)
                            {
                                kattle = true;
```

```
                            }
                        }
                        else if (skel.TrackingState ==
SkeletonTrackingState.PositionOnly)
                        {
                            dc.DrawEllipse(
                            this.centerPointBrush,
                            null,
                            this.SkeletonPointToScreen(skel.Position),
                            BodyCenterThickness,
                            BodyCenterThickness);
                        }
                    }
                }

                // prevent drawing outside of our render area
                this.drawingGroup.ClipGeometry = new RectangleGeometry(new
Rect(0.0, 0.0, RenderWidth, RenderHeight));
            }
        }

        void Gesture_GestureRecognized(object sender, EventArgs e)
        {
            using (DrawingContext dc = this.drawingGroup2.Open())
            {
                dc.DrawRectangle(Brushes.White, null, new Rect(0.0, 0.0,
RenderWidth, RenderHeight / 3));
            }
        }

        /// <summary>
        /// Draws a skeleton's bones and joints
        /// </summary>
        /// <param name="skeleton">skeleton to draw</param>
        /// <param name="drawingContext">drawing context to draw to</param>
        private void DrawBonesAndJoints(Skeleton skeleton, DrawingContext
drawingContext)
        {
            // Render Torso
            this.DrawBone(skeleton, drawingContext, JointType.Head,
JointType.ShoulderCenter);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderLeft);
```

```
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.ShoulderRight);
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderCenter,
JointType.Spine);
            this.DrawBone(skeleton, drawingContext, JointType.Spine,
JointType.HipCenter);
            this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipLeft);
            this.DrawBone(skeleton, drawingContext, JointType.HipCenter,
JointType.HipRight);

            // Left Arm
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderLeft,
JointType.ElbowLeft);
            this.DrawBone(skeleton, drawingContext, JointType.ElbowLeft,
JointType.WristLeft);
            this.DrawBone(skeleton, drawingContext, JointType.WristLeft,
JointType.HandLeft);

            // Right Arm
            this.DrawBone(skeleton, drawingContext, JointType.ShoulderRight,
JointType.ElbowRight);
            this.DrawBone(skeleton, drawingContext, JointType.ElbowRight,
JointType.WristRight);
            this.DrawBone(skeleton, drawingContext, JointType.WristRight,
JointType.HandRight);

            // Left Leg
            this.DrawBone(skeleton, drawingContext, JointType.HipLeft,
JointType.KneeLeft);
            this.DrawBone(skeleton, drawingContext, JointType.KneeLeft,
JointType.AnkleLeft);
            this.DrawBone(skeleton, drawingContext, JointType.AnkleLeft,
JointType.FootLeft);

            // Right Leg
            this.DrawBone(skeleton, drawingContext, JointType.HipRight,
JointType.KneeRight);
            this.DrawBone(skeleton, drawingContext, JointType.KneeRight,
JointType.AnkleRight);
            this.DrawBone(skeleton, drawingContext, JointType.AnkleRight,
JointType.FootRight);
```

```
            // Render Joints
            foreach (Joint joint in skeleton.Joints)
            {
                Brush drawBrush = null;

                if (joint.TrackingState == JointTrackingState.Tracked)
                {
                    drawBrush = this.trackedJointBrush;
                }
                else if (joint.TrackingState == JointTrackingState.Inferred)
                {
                    drawBrush = this.inferredJointBrush;
                }

                if (drawBrush != null)
                {
                    drawingContext.DrawEllipse(drawBrush, null,
this.SkeletonPointToScreen(joint.Position), JointThickness, JointThickness);
                }
            }
        }

        /// <summary>
        /// Maps a SkeletonPoint to lie within our render space and converts
to Point
        /// </summary>
        /// <param name="skelpoint">point to map</param>
        /// <returns>mapped point</returns>
        private Point SkeletonPointToScreen(SkeletonPoint skelpoint)
        {
            // Convert point to depth space.
            // We are not using depth directly, but we do want the points in
our 640x480 output resolution.
            DepthImagePoint depthPoint =
this.sensor.CoordinateMapper.MapSkeletonPointToDepthPoint(skelpoint,
DepthImageFormat.Resolution640x480Fps30);
            return new Point(depthPoint.X, depthPoint.Y);
        }

        /// <summary>
        /// Draws a bone line between two joints
        /// </summary>
        /// <param name="skeleton">skeleton to draw bones from</param>
```

```
        /// <param name="drawingContext">drawing context to draw to</param>
        /// <param name="jointType0">joint to start drawing from</param>
        /// <param name="jointType1">joint to end drawing at</param>
        private void DrawBone(Skeleton skeleton, DrawingContext
drawingContext, JointType jointType0, JointType jointType1)
        {
            Joint joint0 = skeleton.Joints[jointType0];
            Joint joint1 = skeleton.Joints[jointType1];

            // If we can't find either of these joints, exit
            if (joint0.TrackingState == JointTrackingState.NotTracked ||
                joint1.TrackingState == JointTrackingState.NotTracked)
            {
                return;
            }

            // Don't draw if both points are inferred
            if (joint0.TrackingState == JointTrackingState.Inferred &&
                joint1.TrackingState == JointTrackingState.Inferred)
            {
                return;
            }

            // We assume all drawn bones are inferred unless BOTH joints are
tracked
            Pen drawPen = this.inferredBonePen;
            if (joint0.TrackingState == JointTrackingState.Tracked &&
joint1.TrackingState == JointTrackingState.Tracked)
            {
                drawPen = this.trackedBonePen;
            }

            drawingContext.DrawLine(drawPen,
this.SkeletonPointToScreen(joint0.Position),
this.SkeletonPointToScreen(joint1.Position));
        }

        /// <summary>
        /// Handles the checking or unchecking of the seated mode combo box
        /// </summary>
        /// <param name="sender">object sending the event</param>
        /// <param name="e">event arguments</param>
```

```
        private void CheckBoxSeatedModeChanged(object sender, RoutedEventArgs
e)
        {
            if (null != this.sensor)
            {
                if (this.checkBoxSeatedMode.IsChecked.GetValueOrDefault())
                {
                    this.sensor.SkeletonStream.TrackingMode =
SkeletonTrackingMode.Seated;
                }
                else
                {
                    this.sensor.SkeletonStream.TrackingMode =
SkeletonTrackingMode.Default;
                }
            }
        }
    }
}
```

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПРОГРАМНА СИСТЕМА АДАПТИВНОГО УПРАВЛІННЯ ПРИСТРОЯМИ ЗА ТЕХНОЛОГІЄЮ SMART HOME

Виконав: студент групи КП-61, Коломієць Денис Дмитрович

Керівник: доцент кафедри ПЗКС, к.т.н., доцент Сулема Є. С.

Київ – 2020

ПОСТАНОВКА ЗАДАЧІ

Мета проєкту: розробити програмне забезпечення для використання у будинку, облаштованому за технологією Smart Home, для контролю приладів за допомогою жестів через контролер Kinect.

ПОСТАНОВКА ЗАДАЧІ

1. Проаналізувати сучасні методи взаємодії з пристроями за технологією Smart Home.
2. Розробити ПЗ адаптивної системи управління для контролю приладів за допомогою жестів.
3. Протестувати відповідність розробленого ПЗ функціональним вимогам, що визначені у текстовій частині проєкту.

ІСНУЮЧИ МЕТОДИ УПРАВЛІННЯ

amazon alexa

Google ASSISTANT

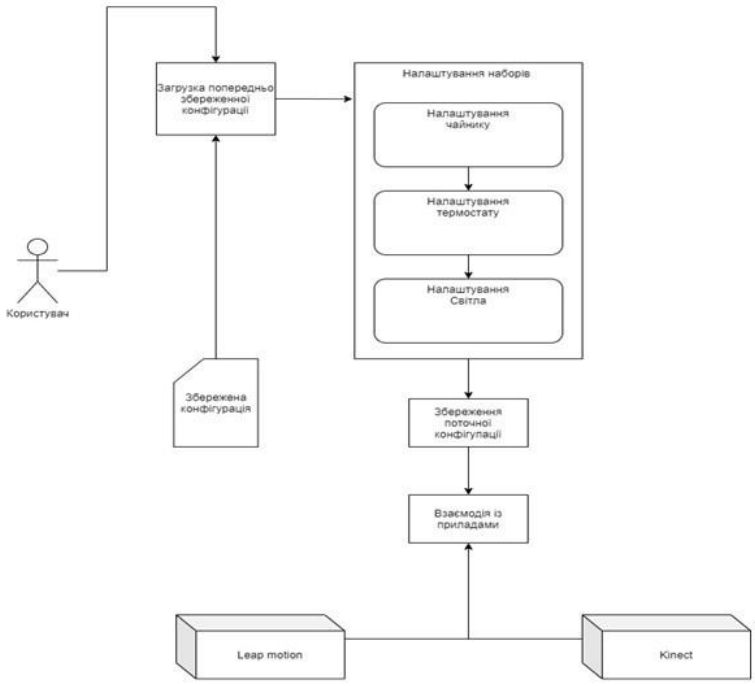
KINECT™

АКТУАЛЬНІСТЬ

1. Кількість приладів, для яких надається можливість використання через зовнішні інтерфейси, постійно зростає.
2. Технологією Smart Home набирає користувачів, тому на її основі розробляються різноманітні системи.
3. Розроблюються методи взаємодії із приладами, що розширює набір зовнішніх пристроїв.

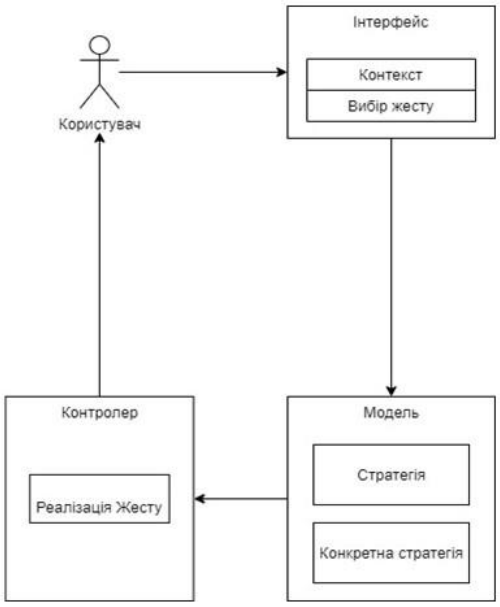
АРХІТЕКТУРА СИСТЕМИ

Діаграма використання



АРХІТЕКТУРА СИСТЕМИ
Діаграма стратегії

АРХІТЕКТУРА СИСТЕМИ
Діаграма реалізованої стратегії



ЗАСОБИ РОЗРОБЛЕННЯ

1. C# – мова програмування, що має офіційні бібліотеки для взаємодії із контролером Kinect і надає можливості для швидкого та зручного розроблення ПЗ.
2. WPF (Windows Presentation Foundation) – офіційний модуль розроблення графічного інтерфейсу користувача для мови C# на платформі Windows.
3. Kinect SDK 1.8 – бібліотека для розроблення застосунків із використанням зчитування жестів з використанням контролеру Kinect.

РОЗРОБЛЕНІ АЛГОРИТМИ ТА ЇХ РЕАЛІЗАЦІЯ

Процедура оброблення жестів активується, коли людина знаходиться у кадрі, який отримується із контролеру жестів, і виконується певна умова конкретного жесту.

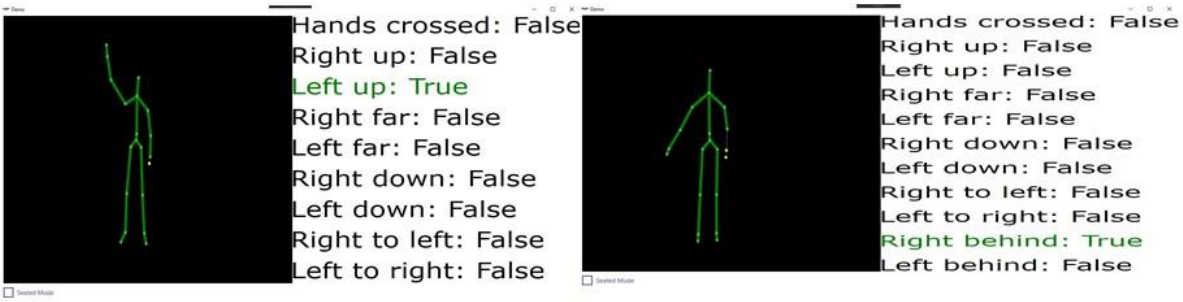
Псевдокод для жесту лівої руки:

```
On Frame Recieved:
    if skeleton is not null:
        if Head.Y_coordinate < Left_Hand.Y_coordinate:
            do(action)

end
```

РОЗРОБЛЕНІ АЛГОРИТМИ ТА ЇХ РЕАЛІЗАЦІЯ

Для реалізації функціональних можливостей, що визначаються ТЗ, було розроблено алгоритми оброблення 6 унікальних жестів і 5 жестів, що є дзеркальним відображенням інших. Таким чином, користувач має 11 жестів для конструкції своїх індивідуальних наборів жестів керування.



РОЗРОБЛЕНІ АЛГОРИТМИ ТА ЇХ РЕАЛІЗАЦІЯ

Жести реалізуються через відстежування кистей обох рук і активуються при потраплянні руки у визначений регіон.

Жест	Умова реєстрації
Занесення руки над головою	Позиція Y кисті більша за позицію голови
Відведення руки від тіла	Різниця позиції X кисті і центра плечей більша за чверть кадру
Заведення руки за плече	Позиція X кисті більша за позицію протилежного плеча
Рука нижче коліна	Позиція Y кисті менша за позицію коліна
Рука за тулубом	Кисть руки не реєструється
Зведення рук навхрест	Позиція X лівої руки більша за праву

КРИТЕРІЇ ОЦІНЮВАННЯ ЯКОСТІ РОЗРОБЛЕНОГО ПЗ

При розробленні критеріїв оцінювання жестів було визначено наступні вимоги:

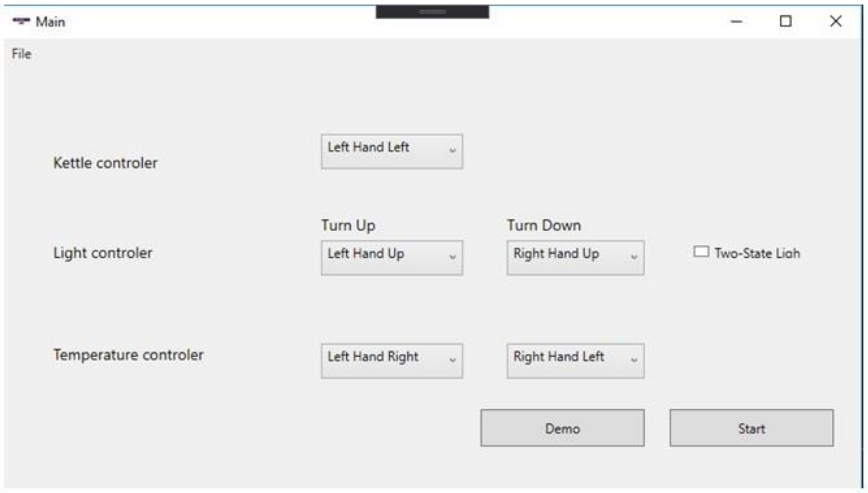
- простота жесту;
- відсутність колізій;
- нетиповість.

Основні критерії до системи є наступними:

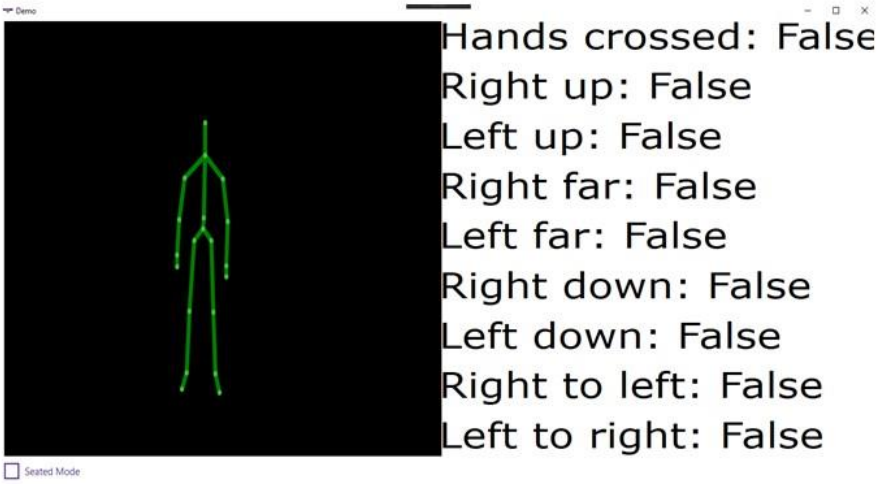
- коректна поведінка граничних значень;
- зручність інтерфейсу.

РОЗРОБЛЕНА СИСТЕМА

РОЗРОБЛЕНА СИСТЕМА
Головне вікно

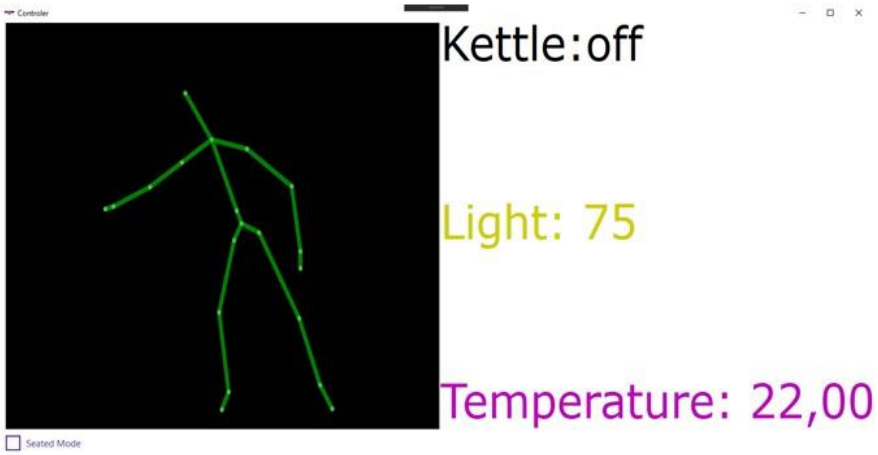


РОЗРОБЛЕНА СИСТЕМА
Демонстраційне вікно



РОЗРОБЛЕНА СИСТЕМА

Вікно контролю



РЕЗУЛЬТАТИ ТЕСТУВАННЯ

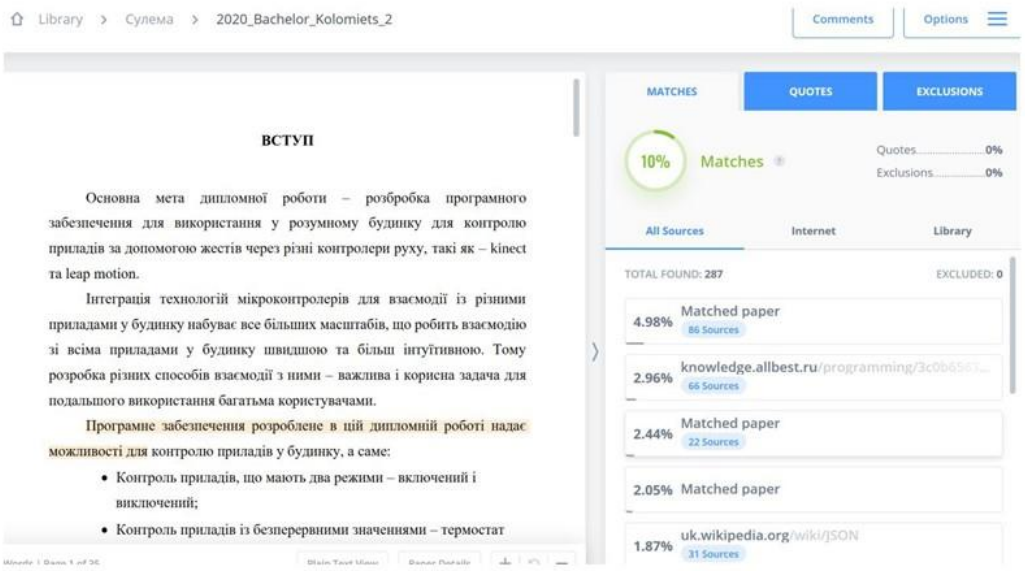
При тестуванні жестів було виявлено, що користувачі з різними типами будови тіла зручніше використовувати різні жести. Тестування жестів було проведене на п’яти користувачах різного віку і типу будови тіла.

Функціональні вимоги виконані повністю і працюють коректно при граничних значеннях.

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Користувач № / Результат тестування	1	2	3	4	5
Виконані жести (максимум 11)	11	11	11	11	9
Кількість колізій при виконанні (максимум 120)	0	0	0	2	3
Рідко виконують жест із набору	+	+	-	+	+

УНІКАЛЬНІСТЬ



ВИСНОВКИ

1. Проаналізовані існуючи методи взаємодії із приладами за технологією Smart Home і виявлено додаткові можливості для розширення функціональних можливостей системи.
2. Запропоновано архітектуру системи, яка дозволить підключення розробленого ПЗ у існуючи системи контролю будинку.
3. Розроблено набір жестів, що використані у даному ПЗ.
4. Розроблено адаптивну систему контролю приладів.
5. Протестовано розроблену систему згідно із встановленими вимогами за допомогою розробленого модулю тестування системи.

Розробку виконано у повному обсязі у відповідності до сформованих вимог, тестування проведено за затвердженою програмою і методикою тестування.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2019 р.

ПРОГРАМНА СИСТЕМА АДАПТИВНОГО УПРАВЛІННЯ ПРИСТРОЯМИ ЗА ТЕХНОЛОГІЄЮ SMART HOME

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Євгенія СУЛЕМА

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Денис КОЛОМІЄЦЬ

2019

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмна система управління індивідуальними пристроями за допомогою пристрою для зчитування жестів «Kinect», розроблений з використанням мови C#, із модулем для розробки віконного інтерфейсу WPF.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) відповідність функціональних можливостей, реалізованих у програмному застосунку до заявлених;
2) коректність поведінки програмного застосунку;
3) можливість виконання розроблених жестів;
4) зручність роботи з системою.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування проводитиметься на рівні «системного тестування», тобто буде перевірено кожен модуль розробленого програмного продукту та зв’язки між ними. Для проведення тестування було обрано метод Black Box Testing. При такому методі перевіряється поведінка програмного продукту при взаємодії користувача із ним.

Використовуються наступні методи:

1) аналіз граничних значень для перевірки коректності виконання при взаємодії;
2) для тестування жестів був розроблений тестовий модуль, що дозволяє протестувати взаємодію із усіма жестами одночасно;
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Коректність роботи системи тестується з використанням таких методів та утиліт:

1) почерговим вибором усіх можливих значень;
2) ручного тестування відповідності реалізованих функціональних вимог до заявлених;
3) тестування жестів через тестовий модуль;
4) тестування працездатності основного модулю системи із використанням усіх жестів в різних комбінаціях;
5) тестування користувачами із різними типами будови тіла;
6) тестування зручності користувацького інтерфейсу опитуванням потенційних користувачів.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2020 р.

ПРОГРАМНА СИСТЕМА АДАПТИВНОГО УПРАВЛІННЯ ПРИСТРОЯМИ ЗА ТЕХНОЛОГІЄЮ SMART HOME

Керівництво користувача

ДП.045440-05-34

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Євгенія СУЛЕМА

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Денис КОЛОМІЄЦЬ

2020

ЗМІСТ

1. Головне вікно

Вигляд головного вікна

У головному вікні користувач може обирати використовувані жести для контролю відповідних приладів, збереження та відтворення сконструйованих наборів, запуск демонстраційного вікна, де користувач може перевірити зручність жестів, запуск вікна контролю для використання приладів.

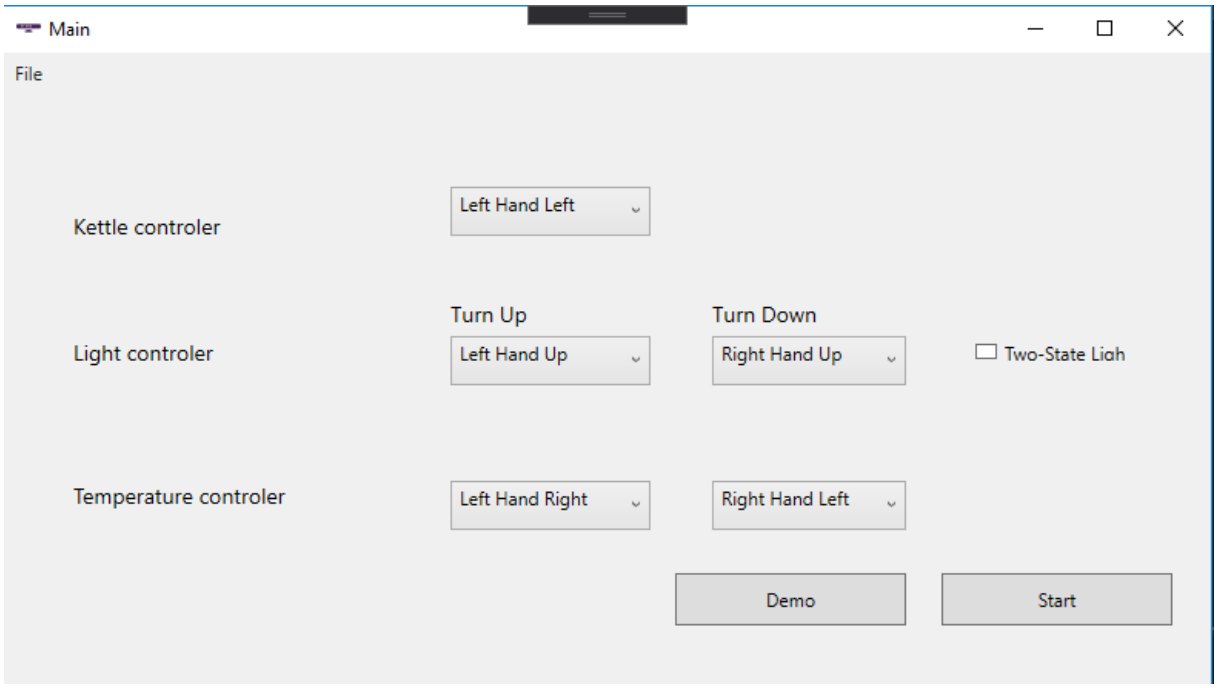


Рис. 1. Головне вікно

Вибір жестів

Користувач може вибрати використовувані жести через взаємодію із списками для відповідних приладів. Користувач може налаштувати жест для включення чайнику, включення і виключення світла через відповідні жести, збільшення і зменшення температури на реєстрації жестів.

Запуск інших вікон

Після налаштування набору, користувач може запустити вікно контролю, натиснувши кнопку «Start». Також користувач може запустити демонстраційне вікно для перевірки зручності жестів.

2. Демонстраційне вікно

Вигляд демонстраційного вікна

У демонстраційному користувач може перевірити жести, що будуть зручні особисто для нього і прийняти рішення щодо конструювання зручних наборів жестів.

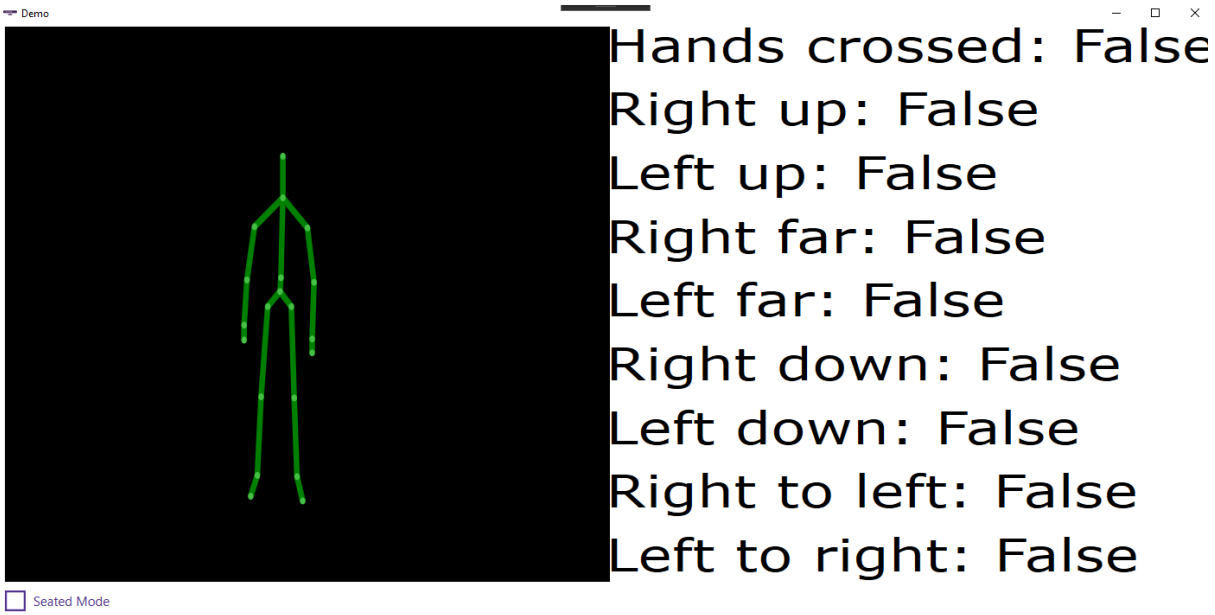


Рис. 2. Вигляд демонстраційного вікна

Вікно поділене на дві частини, де права частина відображає стан кожного індивідуального жесту і відповідно сповіщає користувача при реєстрації конкретного жесту.

3. Вікно контролю

Вигляд вікна контролю

У вікні контролю користувач виконує основну взаємодію із приладами за допомогою сконструйованих наборів жестів. При реєстрації жесту змінюється стан приладу, що також наглядно відображається за рахунок зміни кольору тексту в описанні приладу.

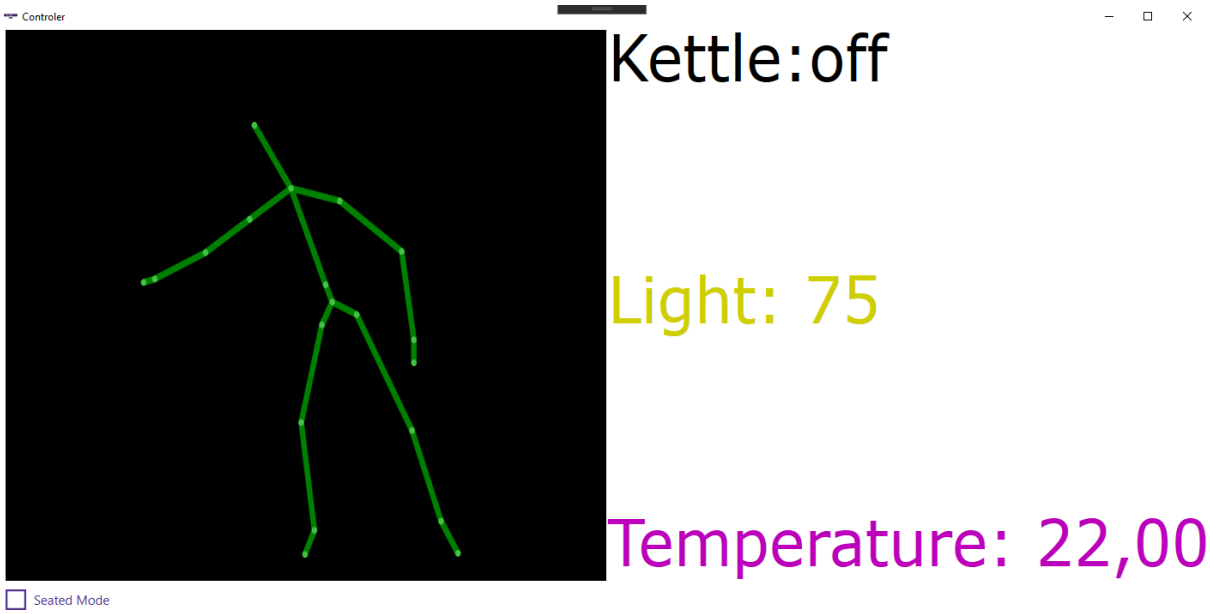


Рис. 3. Вигляд вікна контролю

Вікно контролю відображає віртуальний скелет користувача, що захоплений приладом зчитування Kinect і поведінку та стан приладів, що контролюється за допомогою системи. Таким чином, вікно поділене на дві частини.

У лівій частині вікна відображається скелет, що відслідковується приладом зчитування. Якщо користувач знаходиться у сидячому положенні – він може відключити нижню частину тіла за допомогою поля «Seated mode».