УДК 004.773

DOI: 10.20535/2523-4455.mea.244322

Засоби та методики оцінки ефективності передавання відеопотоку на основі технології GigE Vision з використанням процесору загального призначення

Кужильний[f] О. В., ORCID 0000-0003-2681-5569
Ходнєв Т. А., ORCID 0000-0001-9168-0504
Варфоломєєв[s] А. Ю., к.т.н., ORCID 0000-0002-6990-7140
Кафедра конструювання електронно-обчислювальної апаратури, Факультет електроніки
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського» ROR 00syn5v21
Київ, Україна

Михайленко[s] І. В., к.т.н., ORCID 0000-0002-2655-3119
Міннесотське відділення AELSLAGID
Сент-Пол, Міннесота, США

***Анотація*—У роботі досліджено ефективність реалізації GigE Vision сумісного джерела відеопотоку на обчислювальній платформі, основаній на ARM процесорі загального призначення. Зокрема, для реалізації джерела створено прототип GigE Vision сумісної камери з використанням порівняно розповсюдженого одноплатного комп'ютера Raspberry Pi 4. З використанням програмного інтерфейсу Video4Linux2 розроблено програмну реалізацію процедури захоплення зображень із відеосенсора, підключеного до одноплатного комп'ютера та за допомогою бібліотеки Aravis створено процедуру конвертування і передавання мережею захоплених кадрів у сумісному з технологією GigE Vision форматі. Запропоновано метод вимірювання затримок передачі кадрів каналом Ethernet та проведено відповідні вимірювання. Встановлено, що програмна реалізація GigE Vision сумісної відеокамери на сучасних одноплатних комп'ютерах може вважатися перспективною, в особливості, за подальшого вдосконалення шляхом оптимізації відповідних програмних та/або апаратних складових.**

Ключові слова — відеопотік; затримка передачі; синхронізація часу; Ethernet; GigE Vision

I. Вступ

З метою створення продуктів, які повинні відрізняються гнучкістю та невисокою вартістю, індустрія комп'ютерного зору роками розробляє високоефективні програмні рішення на основі засобів передачі зображень на короткі та середні відстані, таких як Camera Link та LVDS. З цим пов'язаний певний інтерес в галузі машинного зору щодо розробки нових програм з використанням загальнорозповсюджених комунікаційних з'єднань Gigabit-Ethernet [1, 2, 3].

GigE Vision — це технологія передачі відеопотоку на основі Ethernet, зокрема, для задач машинного зору [4, 5]. Дана технологія є відносно економічно-вигідною у використанні, працює на транспортному рівні моделі OSI та дозволяє передавати дані на великі відстані за допомогою стандартизованого кабелю CAT-5e/6 або будь-якого іншого середовища передачі, що підтримується стандартом Ethernet [3]. Як наслідок, GigE Vision підтримує цілий спектр відповідних швидкостей передачі Ethernet каналу, забезпечуючи достатньо високу пропускну здатність, необхідну для потокової передачі зображень з відеосенсорів у реальному часі.

За стандартом GigE Vision версії 2.0, транспортним шаром є протокол прикладного рівня, що має назву GVSP (GigE Vision Streaming Protocol). Для доставки відеоданих GVSP використовує стандартний протокол нижнього, транспортного рівня UDP [6, 7]. Це дозволяє передавати нестиснутий або стиснутий (JPEG, H.264) відеопотік, інформацію про зображення, або інші дані з мінімальними затримками та накладними витратами засобами мережі. Варто зауважити, що взаємодія між передавачем та приймачем будується за асиметричним принципом — основний потік даних надсилається передавачем, тоді як приймач надсилає лише керуючу інформацію. Оскільки протокол UDP не передбачає підтвердження та не гарантує доставки даних, цілісність передачі пакетів,

DOI: 10.20535/2523-4455.mea.244322

GVSP має контролювати приймач, який у разі їх пошкодження чи відсутності робить запит у передавача про повторне надсилання втрачених даних. Цей запит здійснюється через інший протокол — GVCP (GigE Vision Control Protocol). Через GVCP також здійснюється загальне керування GigE Vision камерою, пошук камер у мережі та встановлюється початкове з'єднання.

Питання ефективності передачі відеопотоку каналом Ethernet з використанням технології GigE Vision було розглянуто у статтях [8, 9, 10, 11]. Водночас, в усіх роботах досліджувались особливості реалізації GigE Vision сумісних камер на базі технології FPGA. На підставі аналізу результатів минулих досліджень виникла ідея заміни FPGA процесором загального призначення при розгортанні GigE Vision технології. Водночас, у дослідженні [1] стверджується, що така заміна не є надто доцільною з огляду на низьку енергоефективність процесорів загального призначення, однак зважаючи на час, що пройшов з моменту публікації [1] (приблизно 15 років) ситуація суттєво змінилась. Сучасні системи на кристалі та вбудовані процесори загального призначення стали значно продуктивнішими, енергоефективнішими та мають широкий набір периферії, в тому числі для безпосереднього підключення відеосенсорів та Ethernet контролерів. Враховуючи це, а також нижчу вартість в порівнянні з FPGA, постає питання щодо їх використання як більш дешевих альтернативних рішень для реалізації GigE Vision відеокамер. Цікаво відмітити, що на момент написання даної статті публікацій, окрім роботи [1] на тему програмної реалізації GigE Vision камер з використанням одноплатних комп'ютерів знайдено не було. Причиною цього є імовірно той факт, що на момент появи стандарту GigE Vision на початку 2000 років, використання одноплатних комп'ютерів із процесорами загального призначення не було доцільним і з того моменту реалізацію камер на їх основі всерйоз ніхто не розглядав. З огляду на зазначене *метою* даної статті є дослідження ефективності передавання відепотоку каналом Ethernet на основі технології GigE Vision з використанням саме процесорів загального призначення. При цьому передбачається вирішення задач оцінювання параметрів таких як рівень затримки між передавачем та приймачем, а також рівня навантаження на мережу та процесорні ядра.

II. Реалізація прототипу GigE Vision сумісної камери на основі системи на кристалі з процесорними ядрами загального призначення

Для оцінювання ефективності передавання відеопотоку системою на основі вбудованого процесора загального призначення було розроблено власний прототип GigE Vision сумісної камери. Для цього обрано апаратну платформу, що задовольняє критеріям порівняно низької собівартості, відносно низької споживаної потужності, наявності необхідних периферійних пристроїв та розроблено програмну підтримку первинного захоплення відеопотоку і його подальшого транслювання у Ethernet мережу.

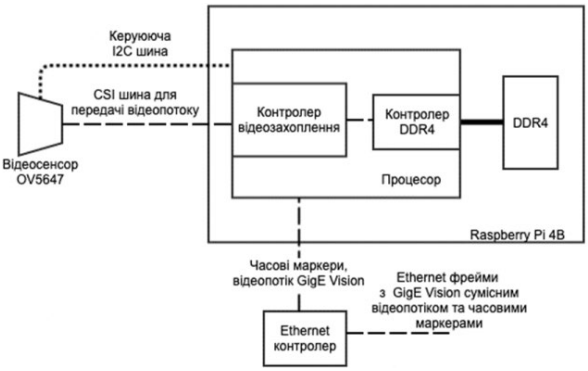

Рис. 1 Структурна організація апаратної частини створюваного прототипу GigE Vision сумісної камери.

Зокрема, прототип GigE Vision камери вирішено створювати на базі апаратної платформи Raspberry Pi 4B [12] та MIPI CSI сумісного відеосенсору OV5647 [13]. Платформа Raspberry Pi 4B була обрана за сукупністю властивостей, зокрема через її доступність, наявність широких обчислювальних можливостей завдяки обчислювально потужному 4-х ядерному процесору BCM2711 на основі ядер Cortex-A72, що працюють на частоті 1,5 ГГц, наявність необхідної периферії: Ethernet контролера з підтримкою стандарту 1000Base-T Gigabit Ethernet, контролера відеозахоплення для отримання зображень з відеосенсору, а також, що важливо, базового пакету програмної підтримки, необхідного для швидкого прототипування.

З огляду на те, що реалізація GigE Vision сумісної камери можлива також і на інших обчислювальних платформах (на основі інших систем на кристалі) щоб за необхідності її можна було легко перенести на ці платформи, прототип створювався з якомога більш апаратно-незалежною структурною організацією, показаною на рис. 1.

III. Реалізація програмної частини дослідження

Програмна частини GigE Vision сумісного прототипу камери складається з двох структурних компонентів: драйверів захоплення відеопотоку та компонента, який конвертує отримані зображення в кадри Ethernet для транслювання у мережу.

A. Підсистема захоплення відеопотоку

З точки зору апаратної складової, обрана система на кристалі має вбудований блок відеозахоплення (BCM2711 VideoCore співроцесор), що виконує отримання кадрів від сенсора та подальше збереження до зовнішньої оперативної пам'яті. Завдяки цьому забезпечується програмна абстракція функціоналу збереження у пам'яті зображень, які далі уніфікованим чином можуть бути передані користувацьким сервісам.

Для отримання кадрів використано відповідну V4L2 (Video4Linux2) підсистему драйверів ядра Linux [14]. З огляду на те, що дана підсистема працює у просторі ядра, V4L2 надає користувацький

DOI: 10.20535/2523-4455.mea.244322

інтерфейс керування у вигляді виклику системної функції **`ioctl()`**. Даний виклик виконується до відеопристрою, що зазвичай ініціалізується у системі як файл з директорії **`/dev`** з іменем **`videoX`**, де **`X`** — номер
зареєстрованого в системі пристрою відеозахоплення. Процедура отримання кадру із пристрою джерела відеопотоку детально описана у відповідній документації Linux [15, 16, 14].

Використовуючи підсистему Video4Linux, було розроблено власну процедуру відеозахоплення, що працює у користувацькому просторі та передає захоплені кадри відеопотоку до підсистеми, що реалізує мережеву складову створюваного прототипу камери. Оскільки, через помилки передавання або дію зовнішніх факторів, потік даних може передаватись мережею з неоднорідним темпом [17]. При цьому, зображення перед відправленням зберігається до кільцевого буферу, кожен елемент якого містить окремий кадр [18]. Розмір буферу може задаватися користувачем як параметр на етапі конфігурації. Організація первинного захоплення кадрів з підсистеми Video4Linux є наступною: підсистема зберігає кадри з простору пам'яті ядра у елементи буферу користувацької пам'яті, інформує про готовність кадру, передає вказівник на початок кадру та його розмір. У результаті, вказівник на буфер та його розмір може бути передано до користувацьких сервісів (в тому числі мережевої підсистеми), які, у свою чергу, матимуть змогу скопіювати вказану кількість байтів, з яких складається зображення.

Підсистема V4L виступає як клас драйверів, що уніфікують механізм для простору користувача з метою взаємодії з відеосенсорами різних виробників за допомогою відповідних апаратно-залежних драйверів. Таким чином, один V4L драйвер може бути використаний для взаємодії з багатьма камерами, причому на різних вбудованих апаратних платформах [19].

B. Мережева підсистема, що реалізує програмний компонент GigE Vision

Бібліотека Aravis є однією, якщо не єдиною відкритою та найбільш повноцінною бібліотекою для роботи з GigE Vision сумісними пристроями [20]. Aravis основана на низькорівневих бібліотеках GLib/GObject та розповсюджується за вільною ліцензією LGPL v2+. Виходячи з цього, було прийнято рішення дослідити дану бібліотеку на предмет її використання як програмного компоненту, що працює у користувацькому просторі Linux та реалізує мережеву складову прототипу камери, що розробляється.

Ідея використання бібліотеки Aravis полягає у тому, що в її складі реалізовано емулятор GigE Vision камери. Підмінивши в даному емуляторі синтетичне зображення на реальне дозволить створити власну GigE Vision сумісну камеру. Така підміна може бути тривіально здійснена шляхом передачі вказівника на кадри, які зберігаються у відеобуфері, отримувані підсистемою відеозахоплення, що була описана у попередньому підрозділі.

Зокрема, створення віртуальної GigE Vision камери, що здатна транслювати реальний відеопотік потребує внесення незначних змін у модулі **`arvfakecamera.c`** бібліотеки Aravis, що включають:

1) ініціалізацію процесу захоплення кадрів — запуск підсистеми відеозахоплення, що заповнюватиме циклічний відеобуфер;
2) вказати функцію, зворотного виклику, що має викликатись при запиті на передавання чергового кадру, для чого присвоїти вказівник на функцію у поле:
   `fake_camera->priv->fill_callback`;
3) реалізувати механізм відображення відеобуферу, тобто надати вказівник на початок кадру, що має бути переданий, шляхом вибору коректного кадру із циклічного буферу та запису його адреси у поле:
   `arv_buffer->priv->data`.

При цьому, бібліотека Aravis бере на себе задачу конвертування зображень у Ethernet фрейми та їх передавання через мережевий сокет з використанням стеку TCP/IP, що реалізується підсистемою ядра Linux.

Структурна організація програмної частини створюваного прототипу камери показана на рис. 2. В наведеній структурній організації підсистема захоплення відеопотоку реалізується блоком «Драйвер відеозахоплення», а мережева підсистема — модифікованим модулем `arvfakecam.c`. Рис. 2 також ілюструє як наведені підсистеми взаємодіють з бібліотеками та компонентами операційної системи Linux.

IV. Дослідження ефективності передавання даних створеним прототипом камери

Дослідження ефективності формування та передавання GigE Vision відеопотоку створеним прототипом містить два аспекти. Перший полягає у оцінюванні затримки передавання кадрів відеопотоку, а другий — у визначенні навантаження на обчислювальну систему, зокрема, процесор, що лежить в основі створюваного прототипу.

A. Методика оцінювання затримок передавання кадрів у дослідженні

Оцінювання затримок передавання кадрів можна здійснити за часовими мітками формування кадру на стороні передавача та часу його отримання приймачем достатньо простим способом [21, 22, 23]:

$$\tau = t_r - t_t,$$

де τ — затримка передавання; t_r — час отримання кадру приймачем; t_t — час початку процедури передавання кадру на стороні передавача.

Не дивлячись на простоту реалізації такої оцінки, з нею пов'язана порівняно суттєва проблема — для того, щоб отримати коректну затримку із різниці часових міток, час на приймачі та передавачі має йти синхронно [22].

DOI: 10.20535/2523-4455.mea.244322

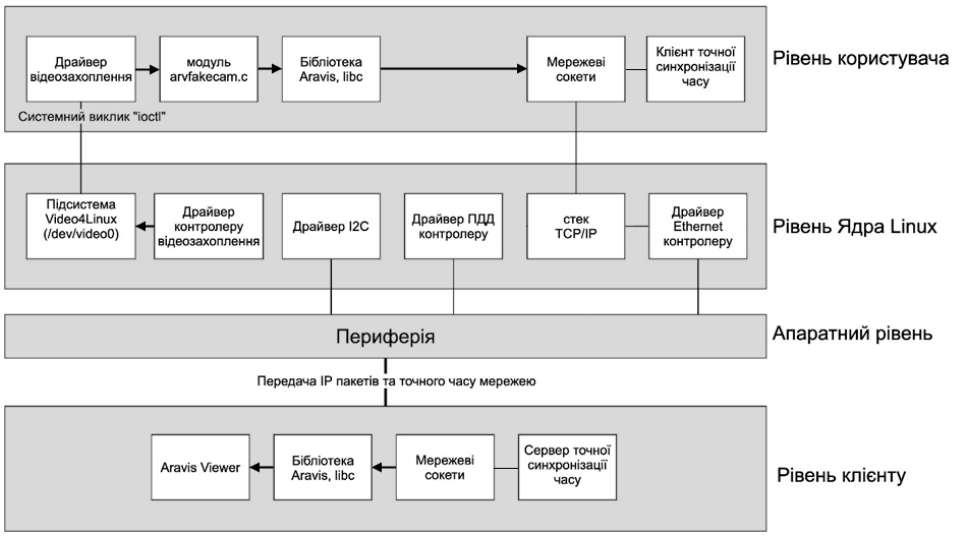

Рис. 2 Структурна організація програмної частини створюваного прототипу GigE Vision сумісної камери. Хоча обмін даними між більшістю компонентів системи є двостороннім, стрілки вказують на напрямок основного потоку інформації

Доречно зазначити, що дана проблема має потенційне рішення при застосуванні протоколу PTP (Precision Time Protocol), визначеного стандартом IEEE-1588. Суть цього протоколу полягає у тому, що вузли, на яких необхідно синхронізувати час, постійно обмінюються сповіщеннями, в яких передається час ведучого вузла (сервера часу), а також здійснюється корегування на величину часу, витраченого на проходження через мережеве обладнання [24]. Ця схема підвищує точність видачі часу споживачам, компенсуючи затримки доставки повідомлень мережею. Стверджується, що для апаратної реалізації протоколу, синхронізація можлива з точністю до 1 мікросекунди, тоді як для програмної реалізації PTP, точність зазвичай не перевищує 100 мікросекунд [25, 26]. Оскільки час формування зображення на відеосенсорі з огляду на реальні величини експозиції перевищують 100 мкс в рази, якщо не на порядки, точності, що забезпечується навіть програмною реалізацією PTP протоколу для оцінювання затримок можна вважати достатньою.

Зауважимо також, що оскільки дослідження проводиться для з'єднання точка-точка, у мережі наявні тільки два пристрої, тому загалом не становить суттєвої різниці, який саме з них буде джерелом точного часу, а який буде його відповідним споживачем. Водночас, в рамках даної роботи сервер точного часу працює на стороні приймача (клієнта).

З практичної точки зору функціонування PTP протоколу в операційній системі GNU/Linux забезпечується програмою ptpd. Її виклик з відповідними аргументами може здійснюватись, наприклад, наступним чином:

- для серверу точного часу:

```
ptpd -C --p2p -i eth0 -M
```

- для клієнту, що має синхронізувати свій час:

```
ptpd -C --p2p -i eth0 -s
```

де ***eth0*** — системне ім'я використовуваного мережевого інтерфейсу Ethernet. Синхронізація часу врахована у структурі створюваного прототипу камери відповідними блоками (див. рис. 2).

B. Результати досліджень

Оцінювання часу затримки передавання кадрів здійснено з використанням методики описаної в попередньому підрозділі для різних роздільних здатностей зображень та показано на рис. 3. На рис. 4 показано відсоток завантаження процесорних ядер передавача (створюваного прототипу камери) при передаванні кадрів з відповідною роздільною здатністю.

З метою виключення впливу на час передавання процедури первинного захоплення кадрів, було проведено додатковий експеримент, в якому здійснювалось надсилання одного незмінного кадру, причому підсистема відеозахоплення на основі V4L залишалася відключеною. Результати експерименту показано на рис. 5 та рис. 6, де наведено затримку передавання та відсоток завантаження ядер процесора відповідно.

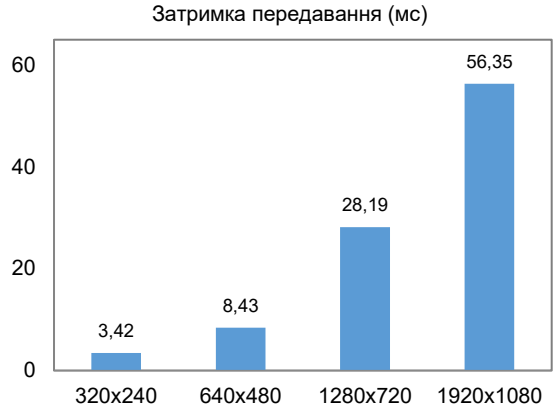

Рис. 3 Усереднена затримка часу передавання кадрів різної роздільної здатності

DOI: 10.20535/2523-4455.mea.244322

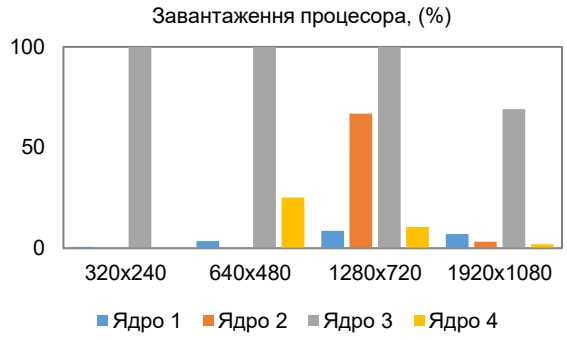

Рис. 4 Середній відсоток завантаження процесорних ядер передавача при передаванні кадрів різної роздільної здатності

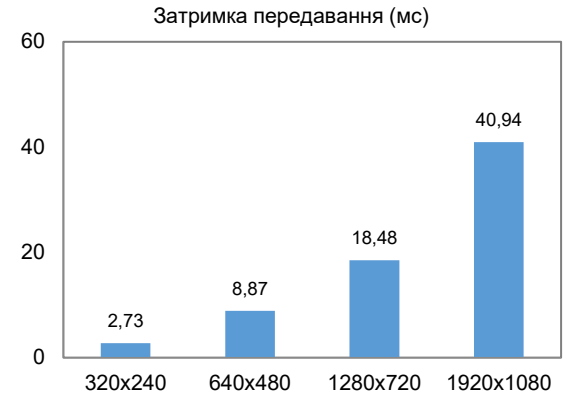

Рис. 5 Усереднена затримка часу передавання кадрів різної роздільної здатності (без виконання процедури захоплення кадрів)

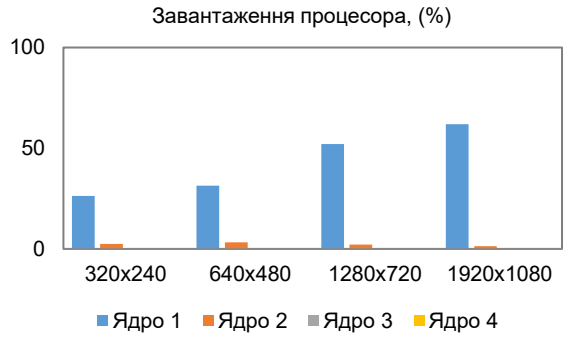

Рис. 6 Середній відсоток завантаження процесорних ядер передавача при передаванні кадрів різної роздільної здатності (без виконання процедури захоплення кадрів)

Порівнюючи діаграми рис. 3 та 5 можна бачити, що час отримання кадрів з підсистемою V4L2 є досить високим та становить близько 30% від загального часу формування та передавання кадрів. Більше того, згідно графіків завантаження процесорних ядер (рис. 4 та 6) близько 50% відсотків припадає на первинне захоплення кадрів. Причину цього можна пояснити тим, що в експерименті для зручності подальшого відображення кадрів на стороні передавача здійснювалось перетворення кадрів у формат YUV. У випадку, якщо це перетворення не виконувати, як це зазвичай відбувається в реальних камерах і передавати від сенсора вихідні (raw) дані минаючи процес перетворення, зазначені затримки можна помітно знизити. З іншого боку з рис. 4 та рис. 6 видно, що постійно завантажується тільки одне ядро процесора, тому навіть за потреби виконання перетворень на тестовій платформі є резерв обчислювальних можливостей, які можуть бути використані шляхом розпаралелювання обчислень.

Водночас, з діаграм рис. 3 та рис. 5 видно, що найбільший внесок у затримку вносить саме передавання кадрів. Як вдалося встановити причиною цього є занадто низьке значення MTU (Maximum Transmission Unit), що може бути реалізоване на тестовій платформі — реальне максимальне підтримуване штатним драйвером MAC Raspberry Pi 4 значення MTU виявилось на рівні лише 2000 байт. Дослідження впливу розміру MTU на час передавання кадрів показало, що, наприклад, при збільшенні MTU з 1400 до 2000 байт час передавання Full HD кадру (роздільною здатністю 1920×1080 пікселів) зменшується приблизно з 40 мс до 32 мс. Щоб спрогнозувати затримки передавання при більших значеннях MTU також було проведено експеримент із залученням персонального комп'ютера, де підтримувались так звані «jumbo-кадри» з максимальним значення MTU на рівні 9000 байт. При цьому, час передавання Full HD кадру при MTU на рівні 8192 байт становив близько 4 мс (тест проводився з обміном через мережевий loopback-інтерфейс, а отже отримана оцінка є дещо оптимістичною). Порівняти зазначені затримки із відповідними затримками для промислових зразків камер складно, оскільки такий параметр у специфікаціях зазвичай не вказують. Водночас, у звіті [5] побіжно згадується, що для серійного зразка камери DALSA Genie HM1400 [27] при налаштуваннях витримки у 100 мкс та передаванні кадрів роздільною здатністю 1400×1024 пікселів на частоті 64 кадрів/с, час отримання кадру (camera readout time) становить близько 15,6 мс. Перераховуючи затримку для створеної програмної реалізації камери пропорційно до роздільної здатності промислового зразка, отримаємо значення затримки на рівні 29 мс, що перевищує час отримання кадру для DALSA Genie HM1400 майже в 2 рази. Не зважаючи на це, на думку авторів створення на одноплатному комп'ютері програмної реалізації камери все ж може вважатись перспективним, оскільки:

- існує можливість подальшої оптимізації по досягненню меншого часу передавання (за рахунок збільшення MTU);
- за умови, якщо не вдасться досягти часу затримки на рівні промислових зразків, то отримана реалізація буде принаймні дешевшою, так як не використовуватиме дороговартісну елементну базу на основі FPGA та все ж зможе знайти застосування у менш чутливих до затримок сферах використання;
- враховуючи, що затримка у 10 мс може вважатись прийнятною для більшості систем комп'ютерного зору [5], при невеликих

DOI: 10.20535/2523-4455.mea.244322

роздільних здатностях (близько 640×480 пікселів) навіть не оптимізована програмна реалізація на одноплатному комп'ютері цілком здатна забезпечити необхідні вимоги (див. рис. 3 та рис. 5).

Висновки

Виходячи з результатів дослідження, можна зробити висновок, що програмна реалізація GigE Vision сумісної камери на одноплатних комп'ютерах є доцільною та може вважатися перспективною. Без додаткових оптимізацій така реалізація дозволяє досягти більш-менш прийнятних результатів затримки для відносно невеликих кадрів (з роздільною здатністю до 640×480 пікселів), де досягається затримка до 10 мс. Встановлено, що найбільш вагомим джерелом затримок є низьке підтримуване значення MTU, тому для кадрів більшого розміру можна рекомендувати вибір обчислювальної платформи, на якій би підтримувалось достатньо велике (рекомендовано до 9000 байт) значення MTU. Крім того, помітний внесок в сумарну затримку передавання може давати процедура перетворення формату зображень. З огляду на це, в рамках подальшої оптимізації, ця процедура може бути перенесена на бік приймача, при цьому передавач буде транслювати вихідні зображення (у форматі raw) або, якщо це дозволяє архітектура платформи, що лежить в основі передавача, перетворення формату може бути розпаралелене і виконуватись на вільних процесорних ядрах.

Перелік посилань

[1] G. Chamberlain, "GigE Vision: standard route to video over IP," 2005. [Online]. Available: https://iebmedia.com/technology/gige-vision-standard-route-to-video-over-ip. [Accessed 25 Oct. 2021].

[2] AIA, EMVA, JIIA, VDMA and CMVU, "Guide to Understanding Machine Vision Standards," 2018. [Online]. Available: https://www.emva.org/wp-content/uploads/FSF-VS-Brochure-2018-A4-full.pdf. [Accessed 27 Oct. 2021].

[3] C. E. Spurgeon and J. Zimmerman, Ethernet: The Definitive Guide, 2nd ed., O'Reilly, 2014. **ISBN**: 9781449361846.

[4] Association for Advancing Automation, "GigE Vision Standards," [Online]. Available: https://www.automate.org/a3-content/vision-standards-gige-vision. [Accessed 14 Dec. 2021].

[5] DALSA_Corporation, "GigE Vision for Real-Time," 2010. [Online]. Available: https://nstx.pppl.gov/nstxhome/DragNDrop/Operations/Diagnostics_&_Support_Sys/D1CCD/GigE_Vision_for_Realtime_MV_11052010.pdf. [Accessed 01 Dec. 2021].

[6] IETF, "RFC 768 — User Datagram Protocol," 1984. [Online]. Available: https://tools.ietf.org/html/rfc768. [Accessed 14 Dec. 2021].

[7] IETF, "RFC 8085 — UDP Usage Guidelines," 2017. [Online]. Available: https://tools.ietf.org/html/rfc808. [Accessed 14 Dec. 2021].

[8] E. Norouznezhad, A. Bigdeli, A. Postula and B. C. Lovell, "A high resolution smart camera with GigE Vision extension for surveillance applications," *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras,* pp. 1-8, 2008. **DOI**: 10.1109/ICDSC.2008.4635711.

[9] O. W. Ibraheem, A. Irwansyah, J. Hagemeyer, M. Porrmann and U. Rueckert, "A resource-efficient multi-camera GigE vision IP core for embedded vision processing platforms," *2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig),* pp. 1-6, 2015. **DOI**: 10.1109/ReConFig.2015.7393282.

[10] V. Marchenko, T. Khodniev and A. Varfolomieiev, "Programno-aparatna realizatsiya videocamery, sumisnoyi zi standartom GigE Vision," *Microsystems, Electronics and Acoustics,* vol. 23, no. 5, pp. 32-37, 2018. **DOI**: 10.20535/2523-4455.2018.23.5.147686.

[11] T. A. Khodniev, M. S. Holub, O. V. Kuzhylnyi, O. M. Lysenko and A. Y. Varfolomieiev, "Accelerated MIPI CSI video stream acquision in tasks of real-time video streaming," *Visnyk NTUU KPI Seriia - Radiotekhnika Radioaparatobuduvannia,* no. 82, pp. 35-43, 2020. **DOI**: 10.20535/RADAP.2020.82.35-43.

[12] Raspberry Pi Foundation, "Raspberry Pi 4 Technical Specifications," [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications. [Accessed 14 Dec. 2021].

[13] Omnivision, "OV5640: color CMOS QSXGA (5 megapixel) image sensor with OmniBSI technology Datasheet," [Online]. Available: https://datasheetspdf.com/datasheet/OV5640.html. [Accessed 14 Dec. 2021].

[14] Linux Kernel Organization, Inc., "The Linux driver implementer's API guide — Video4Linux devices," [Online]. Available: https://www.kernel.org/doc/html/latest/driver-api/media/v4l2-core.html. [Accessed 14 Dec. 2021].

[15] "Media subsystem kernel internal API," [Online]. Available: https://www.kernel.org/doc/html/v4.12/media/media_kapi.html. [Accessed 27 Oct. 2021].

[16] "Video for Linux API capture example," [Online]. Available: https://www.kernel.org/doc/html/v4.12/media/uapi/v4l/capture.c.html. [Accessed 27 Oct. 2021].

[17] J. H. Green, The Irwin Handbook of Telecommunications, 5th ed., McGraw-Hill Education, 2005. **ISBN**: 978-0071452229.

[18] J. Liang and B. Liang, "Effect of delay and buffering on jitter-free streaming over random VBR channels," *IEEE Transactions on Multimedia,* vol. 10, no. 6, p. 1128–1141, 2008. **DOI**: 10.1109/TMM.2008.2001364.

[19] P. R. Schaumont, A Practical Introduction to Hardware/Software Codesign, 2nd ed., Springer US, 2013. **ISBN**: 978-1-4614-3737-6. **DOI:** 10.1007/978-1-4614-3737-6

[20] E. Pacaud, "Aravis — A vision library for genicam based cameras," [Online]. Available: https://wiki.gnome.org/Projects/Aravis. [Accessed 27 Oct. 2021].

[21] A. Kaknjo, M. Rao, E. Omerdic, T. Newe and D. Toal, "Real-Time Secure/Unsecure Video Latency Measurement/Analysis with FPGA-Based Bump-in-the-Wire Security," *Sensors,* vol. 19, no. 13, 2019. **DOI**: 10.3390/s19132984.

[22] R. Mall, Real-Time Systems: Theory and Practice, Pearson, 2006. **ISBN**: 9788131700693.

[23] J. Khatib, A. Munier-Kordon, E. C. Klikpo and K. Trabelsi-Colibet, "Computing latency of a real-time system modeled by Synchronous Dataflow Graph," in *ACM International Conference Proceeding Series*, 2016. **DOI**: 10.1145/2997465.2997479.

[24] S. T. Watt, S. Achanta, H. Abubakari and E. Sagen, "Understanding and Applying Precision Time Protocol," in *Saudi Arabia Smart Grid (SASG)*, Jeddah, 2015. **DOI**: 10.1109/SASG.2015.7449285.

[25] "PTPd," [Online]. Available: https://en.wikipedia.org/wiki/PTPd. [Accessed 27 Oct. 2021].

[26] T. Kovácsházy, "Hardware assisted PTPd home page," 2010. [Online]. Available: http://home.mit.bme.hu/~khazy/ptpd/. [Accessed 27 Oct. 2021].

[27] "DALSA GENIE HM1400 Camera," [Online]. Available: https://www.dalsa-camera-distributor.com/products/genie-hm1400-sv. [Accessed 01 Dec. 2021].

Надійшла до редакції 14 листопада 2021 р.
Прийнята до друку 17 грудня 2021 р.

DOI: 10.20535/2523-4455.mea.244322

UDC 004.773

Means and Methods of Efficiency Estimation of Video Stream Transmission Based on GigE Vision Technology Using Application Processor

O. V. Kuzhylnyi[f], ORCID 0000-0003-2681-5569
T. A. Khodniev, ORCID 0000-0001-9168-0504
A. Y. Varfolomieiev[s], PhD, ORCID 0000-0002-6990-7140
Department of Design of Electronic Computing Equipment, Faculty of Electronics
National technical university of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" ROR 00syn5v21
Kyiv, Ukraine

I. V. Mykhailenko[s], PhD, ORCID 0000-0002-2655-3119
Minnesota Board of AELSLAGID
St. Paul, Minnesota, USA

Abstract*—The paper investigates the possibility of efficient implementation of a GigE Vision compatible video stream source on a computing platform based on a system-on-a-chip with general-purpose ARM processor cores. In particular, to implement the aforementioned video source, a proprietary prototype of a GigE Vision compatible camera was developed based on the Raspberry Pi 4 single-board computer. This computing platform was chosen due to its widespread use and wide community support. The software part of the camera is implemented using the Video4Linux and Aravis libraries. The first library is used for the primary image capturing from a video sensor connected to a single board computer. The second library is intended for forming and transmission of video stream frames compatible with GigE Vision technology over the network. To estimate the delays in the transmission of a video stream over an Ethernet channel, a methodology based on the Precise Time Protocol (PTP) has been proposed and applied. During the experiments, it was found that the software implementation of a GigE Vision compatible camera on single-board computers with general-purpose processor cores is quite promising. Without additional optimization, such an implementation can be successfully used to transmit small frames (with a resolution of up to 640 × 480 pixels), giving a delay less than 10 ms. At the same time, some additional optimizations may be required to transmit larger frames. Namely, a MTU (maximum transmission unit) size value plays the crucial role in latency formation. Thus, to implement a faster camera, it is necessary to select a platform that supports the largest possible MTU (unfortunately, it turned out that it is not possible with Raspberry Pi 4, as it supports relatively small MTU size of up to 2000 bytes). In addition, the image format conversion procedure can noticeably affect the delay. Therefore, it is highly desirable to avoid any frame processing on the transmitter side and, if it is possible, to broadcast raw images. If the conversion of the frame format is necessary, the platform should be chosen so that there are free computing cores on it, which will permit to distribute all necessary frame conversions between these cores using parallelization techniques.

Keywords — video stream; transmission delay; time synchronization; Ethernet; GigE Vision