УДК 004.41
В. П. Судомир, А. М. Луцків канд. техн. наук, доц.
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ПОТОКОВА МОДЕЛЬ ДАНИХ ПРИ ФУНКЦІЙНОМУ ПРОГРАМУВАННІ МІКРОКОНТРОЛЕРІВ

V. P. Sudomyr, A. M. Lutskiv Ph.D., Assoc.

DATA STREAM MODEL AT FUNCTIONAL PROGRAMMING OF MICROCONTROLLERS

Програмне забезпечення для більшості сьогоднішніх мікроконтролерів, зокрема, програми цифрової обробки сигналів, написане імперативними мовами програмування. Імперативні програми швидкі, оскільки вони створені з урахуванням особливостей архітектури обчислювальних пристроїв, але вони не дуже добре відповідають загальній схемі опрацювання сигналу. На відміну від імперативного підходу, функційне програмування і особливо ліниві обчислення у більшій мірі відображають загальний набір операцій над сигналами.

Haskell - це статично типізована функційна мова програмування з лінивими обчисленнями, яка має дуже елегантний та стислий стиль програмування.

Функційний підхід дає змогу використовувати концепцію функцій, у дещо іншому сенсі, аніж для імперативних мов програмування, наприклад, ліниве обчислення означає, що аргументи функції та частини структури даних обчислюються лише в тому випадку, якщо вони потрібні. Таким чином, *потоки опрацювання* можуть містити нескінченно багато елементів. Це не створює проблем оскільки алгоритм здійснює опрацювання лише обмеженої частини цих даних.

Якщо представляти певний сигнал у потоці опрацювання, то не потрібно турбуватися про його довжину, адже можливо зробити її нескінченною і вона буде побудована, лише, настільки, наскільки це потрібно для остаточного застосування або обчислення.

Функційний підхід дозволяє працювати з функціями як і з іншими видами даних. Функції можуть бути аргументами і значеннями інших функцій, так звані функції вищого порядку. Таким чином, у рамках функційної парадигми, циклічні структури не є домінуючими для мови, а користувач може створювати циклічні структури як функції вищого порядку, які приймають тіло циклу як аргумент.

Ключова перевага Haskell - це стислий стиль програмування в поєднанні з статичною перевіркою типу. Слабкою стороною Haskell, на сьогодні, є низька продуктивність. З одного боку програми на мові Haskell добре оптимізуються оптимізаторами, оскільки компілятор може чітко визначити потік даних, а також немає прихованих потоків, які можуть заплутати оптимізатор. З іншого боку, важко спрогнозувати розмір необхідної пам’яті при використанні т.з. “лінивих обчислень”, а відповідно й здійснити оптимізацію коду програми. Велика гнучкість ускладнює створення ефективного коду для певного застосування. Сьогодні існує декілька можливостей налаштування Haskell з метою підвищення ефективності, але ключовим завданням є досягнення як елегантності, так і ефективності.

Можливість програмування мікроконтролерів мовою Haskell надає цікаві можливості використання мови при цифровій обробці сигналів, які отримують та опрацьовують ці мікроконтролери.

На даний момент у вільному доступі є інструмент під назвою «frp arduino», який надає змогу програмувати мікроконтролери Arduino вбудованою предметно

орієнтованою мовою. Ця мова вбудована базується на мові Haskell, й це означає, що програми написані цією мовою, фактично є Haskell-програмами.

Ключовим елементом функційних програм для мікроконтролерів - є потік.

Потік - це скінченна або нескінченна послідовність повідомлень. Якщо M позначає набір повідомлень M^*, набір усіх скінченних послідовностей повідомлень і M^∞ набір усіх нескінченних послідовностей повідомлень для набору всіх потоків по M позначених M^ω, можна визначити:

$$M^\omega = M^\infty \cup M^*$$

Потік містить значення, які змінюються з часом. Прикладом потоку можуть бути дані, які зчитуються з цифрового або аналогового контакту мікроконтролера й опрацьовуються на різних стадіях роботи цього потоку. Таким чином, така побудова програми для мікроконтролера, передбачає захоплення вхідних даних як потоків, проведення над ними обчислень та застосування фільтрів і зберігання результатів виконання потоків (у файл, колекцію, базу даних тощо).

При опрацюванні даних у рамках функційної парадигми з використанням потоків можна умовно виділити наступні типи операцій: породжуючі, трансформаційні та термінальні. Породжуючі операції передбачають отримання даних з деякого вхідного джерела (файлу, колекції, іншого потоку, мережевого з'єднання, входу мікроконтролера тощо) й створення потоку опрацювання. Трансформаційні операції передбачають фільтрування та різноманітні перетворення даних. Трансформаційні операції можуть бути з урахуванням стану (stateful) та без його урахування(stateless). До операцій з урахування стану належать операції агрегування (усереднення, сортування тощо). До термінальних операцій належать операції зберігання даних, виведення на екран, запису в вихідний роз'єм мікроконтролера тощо.

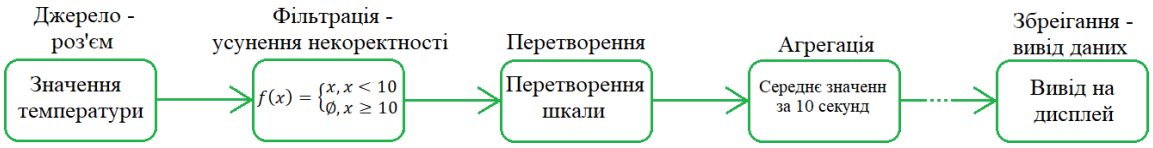

Рисунок 1. Приклад опрацювання даних в потоках.

Література

1. Frp-Arduino [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/frp-arduino/frp-arduino#frp.

2. Функциональное программирование [Електронний ресурс] – Режим доступу до ресурсу: http://pv.bstu.ru/flp/fpLectures.pdf.

3. O'Sullivan B. Real World Haskell: Code You Can Believe In / B. O'Sullivan, J. Goerzen,, D. Stewart., 2008. – 714 с.