РОЗДІЛ IV. ЕЛЕКТРОЕНЕРГЕТИКА, ЕЛЕКТРОТЕХНІКА ТА ЕЛЕКТРОМЕХАНІКА

DOI: 10.25140/2411-5363-2021-4(26)-129-139
УДК 004.453[629.735+004.896]

Володимир Войтенко[1], Роман Єршов[2]

[1]кандидат технічних наук, доцент, доцент кафедри електроніки, робототехніки, автоматики та мехатроніки
Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: volodymyr.voytenko@inel.stu.cn.ua **ORCID:** http://orcid.org/0000-0003-1490-0600
ResearcherID: F-8698-2014. **Scopus Author ID:** 36167678700

[2]старший викладач кафедри електроніки, робототехніки, автоматики та мехатроніки
Національний університет «Чернігівська політехніка» (Чернігів, Україна)
E-mail: roman.d.yershov@gmail.com. **ORCID**: https://orcid.org/0000-0002-0267-2906
ResearcherID: H-1432-2016. **Scopus Author ID:** 57188719994

НАДВИСОКОРІВНЕВЕ ПРОГРАМУВАННЯ СИСТЕМИ ЕЛЕКТРОПРИВОДІВ КВАДРОКОПТЕРІВ ТА АВТОНОМНИХ РОБОТІВ

Розробка програмного проєкту для системи керування електроприводами безпілотного літального апарату (БПЛА) та автономного робота (АР) зазвичай виконується мовами програмування високого або середнього рівня, що збільшує обсяг, складність та час створення коду. У роботі експериментально підтверджена ефективність надвисокорівневого програмування для створення прототипу системи керування електроприводами БПЛА та АР безпосередньо з імітаційної моделі з використанням Embedded Coder® та інструменту STM32 embedded target for MATLAB® and Simulink®.

Ключові слова: *модель електропривода; MATLAB; Simulink; безпілотний літальний апарат (БПЛА); автономний робот; програмне забезпечення системи енергоживлення.*
Рис.: 2. Бібл.: 20.

Актуальність теми дослідження. Безпілотні літальні апарати (БПЛА) та автономні роботи (АР) містять декілька електроприводів, які створюють достатньо складну систему з живленням від акумуляторів. Після моделювання цієї системи [1] настає етап створення прототипу. Оскільки алгоритми керування автономним апаратом переважно реалізуються із застосуванням мікроконтролерів (МК), важливою складовою проєктування є дуже трудомістка стадія розробки програмного забезпечення. Тому мінімізація часу підготовки програмного коду до безпосереднього завантаження у резидентну пам’ять МК є актуальним завданням.

Постановка проблеми. Процес проєктування будь-якої складної технічної системи складається з декількох етапів, впродовж яких треба пов’язувати в часі завершення роботи різних учасників проєкту: фізичних осіб та/або організацій. Електронна система, яку вирішено розробити із застосуванням засобів програмованої логіки (мікропроцесорів, МК та логічних інтегрованих схем, що програмуються), потребує створення програмного коду, обсяг якого варіюється у широких межах. У випадку складної системи електроприводів для автономного апарату цей обсяг може становити кілька тисяч рядків програми мовою високого рівня (C/C++). Написання такого коду, налагодження його, доопрацювання та документування є дуже трудомісткою задачею, яка вимагає долучення кваліфікованих програмістів. Додатковою проблемою є специфіка предметної області та необхідність знань архітектурних особливостей застосовуваних МК. Саме тому потрібні не просто програмісти загального профілю, а ті, хто має великий досвід розробки подібних систем на аналогічних МК.

Проблема посилюється у випадку створення техніки із застосуванням нових принципів та недостатньо досліджених алгоритмів. У цьому випадку пліч-опліч з професійним програмістом має працювати й науковець. Проте робота вимушено стає послідовною в часі, що суттєво затягує процес проєктування. До того ж може виникнути ситуація, коли

в результаті випробувань буде з'ясовано, що обраний МК не в змозі вирішити поставлені перед ним завдання, і треба, можливо, обирати іншу елементну базу, а разом з нею – іншого висококваліфікованого програміста.

Звичайно, найкращим рішенням є поєднання ролі науковця-дослідника та програміста в одній особі. Проте це – ідилія, яка на практиці можлива виключно під час вирішення доволі схожих задач, наприклад, у випадку неглибокої модернізації техніки тощо. За умов зміни МК однаково знадобиться вузький спеціаліст, який витратив чималий часу на засвоєння певної елементної бази. Як аргумент наведемо популярний МК STM32F429, який є представником лінійки процесорів із вбудованими засобами цифрової обробки сигналу, співпроцесором обробки чисел з плаваючою точкою та іншими перевагами, які дозволяють ефективно використовувати його в системах керування електроприводами автономних апаратів. Так, тільки один із важливих документів на цей процесор [2] містить 240 сторінок тексту, які мають бути уважно засвоєні розробником електронної системи. Але для професійної роботи з даним МК потрібно ознайомитися ще з великою кількістю різних документів, з програмними бібліотеками тощо. У випадку зміни цільового процесора на інший, наприклад, на STM32H743ZI [3], який має більш високу тактову частоту та продуктивність, а також – суттєві архітектурні особливості, доведеться додатково вивчити ще мінімум 358 сторінок тексту.

Отже, розробка прототипу системи електроприводів для автономного апарату з новими принципами енергоефективного керування потребує залучення дослідників, які мають бути озброєні якісними інструментами для швидкого створення та зміни програмного коду в залежності від обраного МК.

Аналіз останніх досліджень і публікацій. Зважаючи на низку основних характеристик, найбільш адекватною елементною базою для створення систем керування БПЛА та АР є МК сімейства ARM Cortex [2-4]. Відомі інтегровані середовища налагодження програмного забезпечення систем на базі цих МК використовують мови програмування високого або середнього рівня, що збільшує обсяги та складність програмного коду. Більшість джерел описують ефективне вирішення типових задач програмування та є корисними для навчання та створення типових систем.

Якщо виходити з кінцевої мети – завантаження програмного коду до резидентної пам'яті МК, то цьому процесу має передувати створення програми (найчастіше – у шістнадцятковому форматі Intel) за допомогою того чи іншого інструментального програмного забезпечення (ПЗ).

Для тих МК, що розглядаються в цій роботі, таким інструментом може бути одне з наступних інтегрованих середовищ налагодження ПЗ:

• EWARM від IAR [5];
• MDK-ARM від Keil [6];
• STM32CubeIDE від STMicroelectronics [7].

У всіх випадках програмний проєкт створюється з використанням мов програмування C/C++. Мова програмування взагалі – це штучний формалізм, за допомогою якого можна виразити алгоритми [8]. Відповідно до сучасної класифікації багаточисельний набір існуючих мов програмування розрізняють за поколіннями, за парадигмами програмування, за рівнями та іншими ознаками. У контексті цієї роботи найбільш цікавим є розрізнення мов програмування за рівнем наближення до людини або до машини. За цим критерієм розглядають мови програмування:

- низького рівня;
- високого рівня;
- надвисокого рівня.

Іноді множину використовуваних мов високого рівня розбивають на два підкласи: саме високого рівня, а також – середнього рівня. Ті мови програмування C/C++, що нас цікавлять, належать, формально, до мов програмування високого рівня, проте використовуваний у них рівень абстракції доволі невисокий: неспроста мова С на початку створення вважалася різновидом макросасемблера. Зважаючи на досить низький рівень цих мов, виникає проблема: для створення прототипу системи керування електроприводами автономного апарату потрібна дуже трудомістка розробка програми мовою середнього рівня. Водночас прототипування вимагає чималої кількості спроб, у тому числі, – з різними МК.

Виділення недосліджених частин загальної проблеми. Існує велика кількість публікацій та прикладів ефективного програмування мовами C/C++, наприклад [9-12]. Однак ці та інші джерела сконцентровані передусім на вирішенні типових задач, а імплементація просунутих алгоритмів керування потребуватиме пошуку (розробки), налаштування (налагодження) та тестування специфічних проблемно-орієнтованих бібліотек. Витрати часу на створення цих бібліотек і програм можуть стати марними у випадку переходу на іншу елементну базу. Проте, як було зазначено в [1], впровадження алгоритмів енергоефективного керування системами електроприводів БПЛА та АР доцільно розпочинати з моделювання, що дозволяє заощадити час та матеріальні ресурси. Наявність якісної імітаційної моделі системи електроприводів у середовищі комп'ютерного моделювання MATLAB® та Simulink® [13] відкриває можливості застосування мови програмування надвисокого рівня. Завдяки наявності в цьому середовищі засобів генерації програмного коду (Embedded Coder® [14]), фактично, відкривається можливість застосування мов надвисокого рівня для розробки прототипу системи керування електроприводами автономного апарату.

Мета статті. Метою статті є аналіз наявних можливостей використання мов програмування надвисокого рівня в середовищі комп'ютерного моделювання MATLAB® та Simulink® для створення програмного проєкту прототипу системи керування БПЛА або робота з автономним живленням, що дозволяє суттєво скоротити час проєктування.

Можливості Embedded Coder. Створення програмного коду з налагоджених скриптів та моделей у середовищі комп'ютерного моделювання MATLAB® та Simulink® здійснюється за допомогою таких інструментів, як MATLAB Coder™ і Simulink Coder™. Проте створений таким чином код призначений для додатків, що виконуються на персональному комп'ютері поза межами MATLAB. Системи керування БПЛА та автономних роботів наразі реалізуються на основі вбудованих процесорів, архітектура яких суттєво відмінна від такої для потужних процесорів персональних комп'ютерів. Embedded Coder® – це специфічний інструмент MATLAB, що генерує читабельний, компактний і швидкий код мовами програмування С і С++ саме для вбудованих процесорів, які використовуються в масовому виробництві [14].

Embedded Coder®, фактично, розширює MATLAB Coder™ і Simulink Coder™, та, завдяки якісній оптимізації програми, забезпечує тонкий контроль згенерованих функцій, файлів і даних. Така оптимізація покращує ефективність коду та полегшує інтеграцію із раніше розробленими програмами, типами даних та параметрами калібрування. Для створення виконуваного файлу потрібно підключити сторонній інструмент розробки. Цей підхід, врешті-решт, дозволяє пришвидшити розробку вбудованої системи, особливо, – використовуючи плату прототипування.

Для налаштування параметрів генерації коду Embedded Coder® використовується або додаток MATLAB Coder для MATLAB®, або Embedded Coder Quick Start для Simulink. Кожен параметр також можна налаштувати безпосередньо за допомогою команд і скриптів MATLAB.

Додаток MATLAB Coder дозволяє:

– генерувати програмний код для файлів і функцій MATLAB;
– вибирати процесор і код, що генерується на виході;
– налаштовувати оптимізацію Embedded Coder.

Використовуючи Embedded Coder Quick Start для Simulink, можна:
– згенерувати програмний код для моделей і підсистем Simulink;
– вибрати процесор і код, що генерується на виході;
– застосувати Embedded Coder для оптимізації оперативної пам'яті або швидкості виконання.

Embedded Coder використовує об'єкти конфігурації та системні цільові файли, щоб перевести розроблений код MATLAB та моделі Simulink у вихідний код і виконувані файли високої якості. Крім того, MathWorks і сторонні розробники пропонують доповнення MATLAB, які розширюють Embedded Coder для підтримки певного обладнання, включаючи ARM®, Intel®, NXP™, STMicroelectronics® і Texas Instruments™ [15].

Simulink Embedded Coder значно розширює структуру виконання в реальному часі. За замовчуванням код може виконуватися під керуванням операційної системи реального часу (RTOS) або без неї, а також – в однозадачному, багатозадачному, багатоядерному або асинхронному режимі.

Embedded Coder генерує розширювану основну програму на основі інформації, яку надає програміст для розгортання коду у використаному середовищі реального часу. Ця можливість дозволяє генерувати та будувати повний налаштований виконуваний файл з розробленої моделі.

Генерація одношвидкісного або багатошвидкісного коду базується на періодичних часах вибірки, зазначених в моделі. Для багатошвидкісних і багатозадачних моделей інструмент використовує стратегію, яка називається групуванням швидкості, і яка генерує окремі функції для завдання базової швидкості та для кожного завдання з нижньою частотою в моделі. Можна також використовувати моделювання Simulink Concurrent Execution для створення багатопоточного коду і багатоядерної обробки.

Перевірка результатів виконання коду виконується за допомогою двох механізмів:
–програмне забезпечення в контурі (software-in-the-loop, SIL);
–процесор в контурі (processor-in-the-loop, PIL). При цьому використовується і вбудована апаратна платформа, і режими моделювання Simulink або блоки S-функцій.

Simulink Test™ і Simulink Coverage допомагають автоматизувати виконання тесту та порівняння результатів з результатами моделювання оригінальної моделі. Аналіз покриття структурного коду для вимірювання повноти тестування можна виконати за допомогою Simulink Coverage або за допомогою інтеграції з інструментами сторонніх розробників. Аналіз профілювання коду дозволяє побачити час виконання на хості або цільових процесорах.

Використання STM32 embedded target for MATLAB® and Simulink®. У [16] представлені систематизовані дані про оглядові дослідження різноманітних технологій візуального програмування з точки зору охоплення вимірів. Незважаючи на різницю підходів візуального програмування, більшість інструментів використовують високорівневі абстракції, щоб приховати деталі реалізації, і застосовують схожі стилі взаємодії, такі, як пряме маніпулювання (у формі перетягування), а також вибір у меню. Відзначено певні спільні риси інструментів однієї предметної області.

Інструмент розробки програмного забезпечення STM32 Embedded Target [17] дозволяє швидко розгортати моделі додатків у MATLAB і Simulink на МК STM32. Використовуючи тестування процесора в контурі (Processor In the Loop, PIL), можна перевірити та профілювати результати виконання програми на МК STM32 у порівнянні з поведінкою

Simulink®-моделі. Для запуску моделей додатків Simulink з конфігурацією PIL використовується канал зв'язку USART. STM32 Embedded Target надає бібліотеку блоків Simulink® для периферійних пристроїв STM32 і дозволяє автоматично генерувати код С на основі бібліотек HAL. STM32-MAT/TARGET дозволяє контролювати програми, що працюють на STM32, за допомогою зовнішнього режиму. Користувач контролює параметри програми, яка працює на STM32, і візуалізує результати з Simulink®. Результатом є звіт про створення коду та звіт про профілювання виконання коду. Автоматично згенерований код далі можна побудувати в середовищі відлагодження ПЗ та завантажити до резидентної пам'яті МК.

Після інсталяції STM32 embedded target for MATLAB® and Simulink® у вікні браузера бібліотек Simulink (Simulink Library Browser) з'являється група Target Support Package – STM32 Adapter. У цій групі можна побачити моделі таких Simulink-блоків.

STM32 Configuration model. Блок, який використовується для встановлення конфігурації STM32 за допомогою інструмента STM32CubeMx. Цей блок є обов'язковим для всіх додатків STM32 Simulink і має бути використаний один раз.

ADC. Блок, що використовується в моделі для отримання значення аналого-цифрового перетворення (АЦП) STM32. Залежить від конфігурації STM32CubeMx з відповідними каналами та з конфігураціями переривань АЦП.

CAN Library. Група блоків для роботи з інтерфейсом CAN.

DAC. Блок, який використовується в моделі для вибору цифро-аналогового перетворювача (ЦАП).

GPIO Library. Група блоків для роботи з портами введення-виведення загального призначення (GPIO): читання стану ніжки, запису 0, 1 або зміни значення ніжки порту STM32, а також вибору зовнішнього переривання. Конфігурація портів і контактів GPIO виконується за допомогою STM32CubeMx.

HRTimer Library. Блоки, що використовується в моделі для перегляду конфігурації STM32 HRTimer, дозволу переривань, отримання значення регістра порівняння та запуску подій.

I2S Library. Блоки на основі бібліотеки HAL, що використовуються в моделі для встановлення розміру буфера передачі даних інтерфейсу I2S, отримання вхідних даних, керування прямим доступом до пам'яті.

IWDG (Independent WatchDoG). Блок, який використовується в моделі для ініціювання скидання МК, коли оновлення сторожового таймера не виконується до закінчення тайм-ауту.

OTHER (Memory). Повертає покажчик на дані зчитування периферійного пристрою SAI з будь-яким зміщенням або без нього.

REG Access. Блок, що використовується в моделі для генерування рядка коду С для доступу до периферійного регістра STM32.

SAI (Serial audio interface). Бібліотека генерації коду для послідовного звукового інтерфейсу STM32.

SPI. Модель, яка використовується для створення процедури SPI STM32 на основі бібліотеки HAL.

Timers. Блок, який використовується в моделі для настроювання до 4 каналів таймерів STM32, які були налаштовані у STM32CubeMx. Також дозволяє підключитися до керування перериваннями.

USART Library. Блоки, що використовуються в моделі для отримання та надсилання байтів за допомогою USART/UART, налаштованого у STM32CubeMx.

З використанням певних описаних блоків створимо просту модель для демонстрації роботи з портами введення-виведення загального призначення GPIO (рис. 1). До двох

ліній кожного з портів C та B у лабораторному стенді підключені кнопки SB1…SB4. Модель наказує стан даних портів повторювати на виводах Pin0, Pin1, Pin3, Pin4 порту GPIOA, до яких підключені світлодіоди. Отже, натискання на кнопки має призводити до зміни інформаційної моделі. Логічний зв'язок між станом кнопок та окремими виводами порту GPIOA створюється саме всередині Simulink-моделі шляхом ліній сполучення.

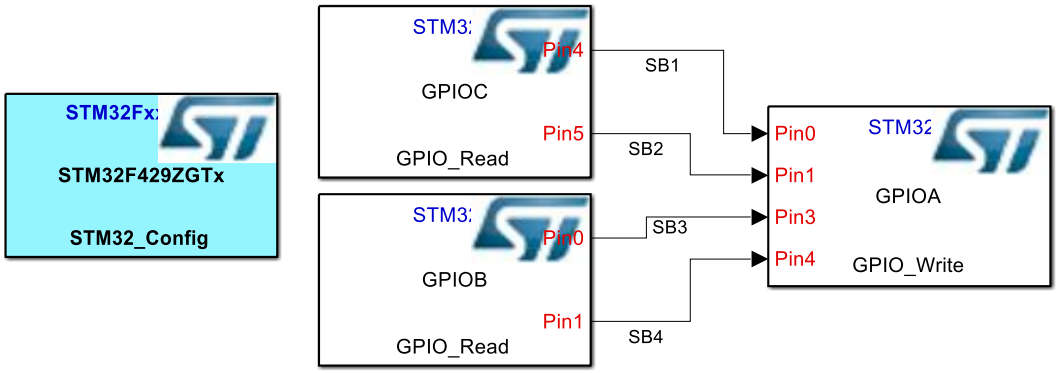

Рис. 1. Модель для перевірки працездатності генератора коду

Оскільки в цій моделі не використано жодного блоку Simulink, запуск симуляції не призводить до якихось дій у вікні моделювання. Подальші дії мають бути пов’язані із конфігуруванням МК за допомогою генератора коду ініціалізації STM32CubeMX [17]. У результаті роботи цієї програми створюється файл конфігурації МК, який має бути розміщеним у тій самій директорії, де зберігається файл Simulink-моделі. Цей файл сам по собі достатній, щоб одразу на його основі створити програмний проєкт та передати його в середовище відлагодження програмного забезпечення МК (згадані вище EWARM, MDK-ARM або STM32CubeIDE) для подальшої побудови програми, що має виконуватися. Проте в такій програмі буде все необхідне для початкового налаштування МК, окрім головного: логічного зв’язку між кнопками та виводами порту GPIOA.

Запуск генерації коду з вікна Simulink-моделі розпочинає процес, в результаті якого Simulink Embedded Coder створює окрему директорію, в якій з’являються запропоновані для побудови проєкту файли. Дуже зручним є створення ретельної документації проєкту у вигляді сукупності пов’язаних між собою гіперпосиланнями html-файлів. Залишаючись всередині середовища MATLAB-Simulink, можна скористатися браузером проєкту, за допомогою якого наявний швидкий доступ до звіту за результатами генерації коду, у тому числі, – до окремих функцій та змінних.

Логіку роботи програми можна побачити в автоматично згенерованому C-файлі, ім’я якого повторює ім’я моделі. Так, для нашого прикладу Simulink Embedded Coder створив наступну структуру для роботи з кнопками:

```
/* Block signals (default storage) */
typedef struct {
  boolean_T SB1;                /* '<Root>/GPIO_Read' */
  boolean_T SB2;                /* '<Root>/GPIO_Read' */
  boolean_T SB3;                /* '<Root>/GPIO_Read1' */
  boolean_T SB4;                /* '<Root>/GPIO_Read1' */
} B;
```

Нижче в програмі згенерований доступ до елементу цієї структури у вигляді:

```
/* Block signals (default storage) */
  B rtB;
```

Для одночасної роботи з 16 розрядами порту GPIOA автоматично додана змінна

```
/* GPIOA output mask value definition. */
  uint16_t GPIOA_maskWrite;
```

З головної функції програми відбувається циклічний виклик наступної функції:

```
/* Model step function */
void Test1_step(void)
{
  /* S-Function (GPIO_Read): '<Root>/GPIO_Read' */
  {
    rtB.SB1 = (boolean_T)HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_4);
    rtB.SB2 = (boolean_T)HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_5);
  }

  /* S-Function (GPIO_Read): '<Root>/GPIO_Read1' */
  {
    rtB.SB3 = (boolean_T)HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_0);
    rtB.SB4 = (boolean_T)HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_1);
  }

  /* S-Function (GPIO_Write): '<Root>/GPIO_Write' */
  {
    /* Set GPIOA output mask value. */
    GPIOA_maskWrite = GPIOA->ODR;
    GPIOA_maskWrite &= 0xFFE4 ;
    GPIOA_maskWrite |= (uint16_t)rtB.SB1 << 0;
    GPIOA_maskWrite |= (uint16_t)rtB.SB2 << 1;
    GPIOA_maskWrite |= (uint16_t)rtB.SB3 << 3;
    GPIOA_maskWrite |= (uint16_t)rtB.SB4 << 4;

    /* Write GPIOA input value */
    GPIOA->ODR = (uint16_t)GPIOA_maskWrite;
  }
}
```

Таким чином, ми бачимо, що S-функція зчитування ніжки МК *GPIO_Read* (два блоки на рис. 1: GPIOB та GPIOC) генератором коду автоматично перетворюється на два дворядкові фрагменти, які базуються на викликах релевантної для цього МК функції бібліотеки HAL (HAL_GPIO_ReadPin()), які заповнюють елементи структури логічними змінними, зчитаними з відповідних ліній портів. S-функція виведення даних у порт МК *GPIO_Write* (блок GPIOA на Рис. 1) автоматично перетворена на функцію маскування бітів регістру вихідних даних МК ODR порту GPIOA за допомогою логічних функцій «І» та «АБО» в залежності від зчитаного раніше стану кнопок.

На рис. 2 показана значно більш складна модель електроприводу автономного апарата [1], розширена достатньо простими функціями, що має виконувати МК (замикання контуру керування, формування завдання та індикація керуючої інформації) з використанням тих же самих виводів портів МК загального призначення, що й у прикладі на рис. 1.

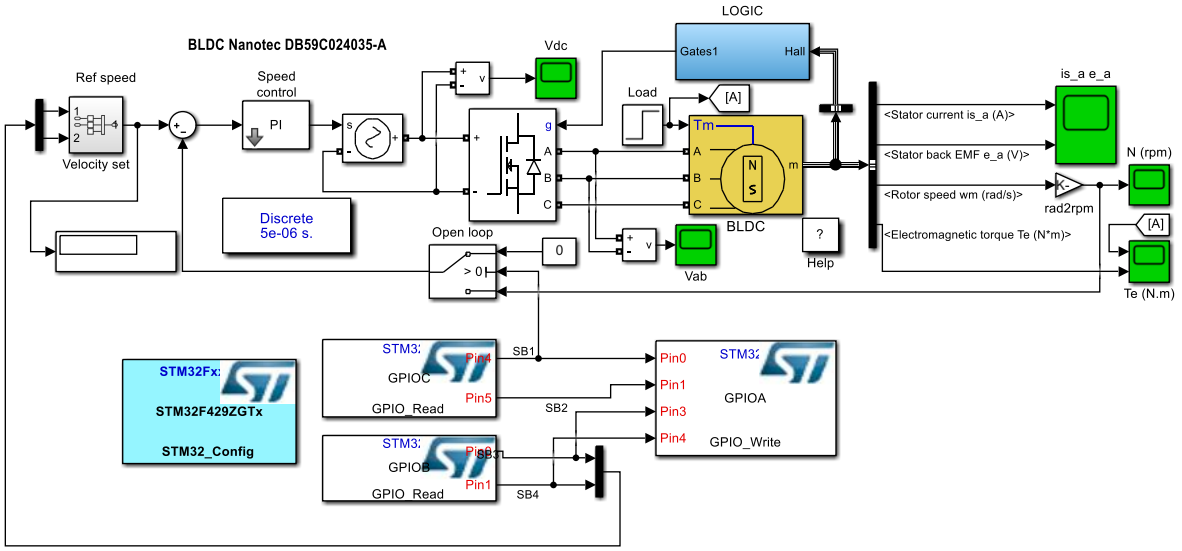

Рис. 2. Модель електропривода автономного апарата

Генерація коду за допомогою Simulink Embedded Coder на комп'ютері під керуванням 64-розрядної операційної системи Windows 11 Home (версія 21H2, збірка 22000.348) з процесором Intel® Core™ i7-9750H @ 2,60 ГГц та оперативною пам'яттю 16 Гб зайняла 23,569 с. При цьому було згенеровано 4 вхідних файли проєкту мовою С загальним обсягом у 1628 рядків коду, 4 заголовкові файли (601 рядок), makefile (327 рядків), а також численні файли документації проєкту.

Усі вхідні файли мають коментарі, вказівки на місцях, де рекомендоване вставлення тих чи інших фрагментів коду користувача, що полегшує орієнтацію у програмі та дозволяє доопрацьовувати її в подальшому. Звичайно, частину блоків (індикація, живлення, силові каскади та ін.) було автоматично виключено зі створеної програми, оскільки немає жодного сенсу покладати на МК вирішення певних задач моделі.

Висновки. Проведені дослідження експериментально підтверджують можливість і ефективність використання наявних інструментів програмування надвисокого рівня для розробки систем керування електроприводами БПЛА та автономних роботів на базі вбудованих МК. Час створення великого за обсягом вхідного коду (порядку двох тисяч рядків мовою С), який автоматично будується за допомогою Simulink Embedded Coder з використанням перевіреної Simulink-моделі електроприводу та може стати основою кінцевого програмного проєкту, складає десятки секунд.

Подальші зусилля доцільно сконцентрувати на прямому портуванні автоматично створеного коду до середовища розробки програмного забезпечення [19], а також – на дослідженні особливостей використання методу PIL (Processor-In the-Loop), що, для відносно нешвидких процесів в системі автономного електроживлення, може дозволити проводити експерименти в реальному часі безпосередньо з середовища MATLAB-Simulink і, таким чином, підвищити ефективність і результативність моделювання складних систем [20].

Список використаних джерел

1. Войтенко, В. Моделі елементів системи електроприводів квадрокоптерів та автономних роботів / В. Войтенко, Р. Єршов // Технічні науки та технології. – 2019. – № 3. – С. 175-187.

2. STM32F429ZG. High-performance advanced line, ARM Cortex-M4 core with DSP and FPU, 1 Mbyte Flash, 180 MHz CPU, ART Accelerator, Chrom-ART Accelerator, FMC with SDRAM, TFT [Electronic resource]. – Access mode: https://www.st.com/en/microcontrollers-microprocessors/stm32f429zg.html.

3. STM32H743ZI. High-performance and DSP with DP-FPU, Arm Cortex-M7 MCU with 2MBytes of Flash memory, 1MB RAM, 480 MHz CPU, Art Accelerator, L1 cache, external memory interface, large set of peripherals [Electronic resource]. – Access mode: https://www.st.com/en/microcontrollers-microprocessors/stm32h743zi.html.

4. ARM® Cortex®-M7 Processor Technical Reference Manual [Electronic resource]. – Access mode: https://developer.arm.com/documentation/ddi0489/f.

5. IAR Embedded Workbench for Arm [Electronic resource]. – Access mode: https://www.iar.com/ewarm.

6. MDK Microcontroller Development Kit [Electronic resource]. – Access mode: https://www2.keil.com/mdk5/.

7. STM32CubeIDE. Integrated Development Environment for STM32 [Electronic resource]. – Access mode: https://www.st.com/en/development-tools/stm32cubeide.html.

8. Gabbrielli, M. (2010). *Programming Languages: Principles and Paradigms* / M. Gabbrielli, S. Martini. [Electronic resource]. – Access mode: http://websrv.dthu.edu.vn/attachments/newsevents/content2415/Programming_Languages_-_Principles_and_Paradigms_thereds1106.pdf.

9. Васильєв, О. Програмування на С++ в прикладах і задачах : навч. посіб. / О. Васильєв – К. : Ліра-К., 2017. – 382 с.

10. Banahan, M. *The C Book – Table of Contents* [Electronic resource] / M. Banahan, D. Brady, M. Doran – Access mode: https://publications.gbdirect.co.uk//c_book/the_c_book.pdf.

11. Beej's Guide to C Programming [Electronic resource]. – Access mode: https://beej.us/guide/bgc/pdf/bgc_usl_c_2.pdf.

12. Object-Oriented Programming with ANSI-C [Electronic resource]. – Access mode: https://www.cs.rit.edu/~ats/books/ooc.pdf.

13. Mathworks: Products and Services [Electronic resource]. – Access mode: https://www.mathworks.com.

14. Embedded Coder: Generate C and C++ code optimized for embedded systems. [Electronic resource]. – Access mode: https://www.mathworks.com/products/embedded-coder.html.

15. Войтенко В. П. Сигнальный процессор в системе управления повышающим квазирезонансным преобразователем / В. П. Войтенко, М. А. Хоменко // Технічна електродинаміка. Тем. випуск "Силова електроніка та енергоефективність", 2012. – Ч. 2. – С. 101-106.

16. Characterizing Visual Programming Approaches for End-User Developers: A Systematic Review. [Electronic resource] / M. A. Kuhail, S. Farooq, R. Hammad, M. Bahja // IEEE Access. – 2021. – № 9. – Pp. 14181-14202. – Access mode: https://ieeexplore.ieee.org/document/9320477.

17. STM32 embedded target for MATLAB and Simulink with PIL and external mode processing (RN0087) [Electronic resource]. – Access mode: https://www.st.com/en/development-tools/stm32-mat-target.html#documentatio.

18. STM32Cube initialization code generator [Electronic resource]. – Access mode: https://www.st.com/en/development-tools/stm32cubemx.html.

19. Hong, Ye Port the Generated ARM Cortex-M CRL Code from MATLAB to KEIL μVision IDE [Electronic resource] / Ye Hong. – Access mode: https://www.mathworks.com/matlabcentral/fileexchange/48809-port-the-generated-arm-cortex-m-crl-code-from-matlab-to-keil-vision-ide, MATLAB Central File Exchange.

20. Khomenko, M. Neural Network based Optimal Control of a DC Motor Positioning System / M. Khomenko, V. Voytenko, Y. Vagapov // Automation and Control. – Vol. 7, Nos. ½. – Pp. 83-104.

References

1. Voytenko, V., Yershov, R. (2019). Modeli elementiv systemy elektropryvodiv kvadrakopteriv ta avtonomnykh robotiv [Models of Elements of The Electric Drive System of The Quadcopters and Autonomous Robots]. *Technical Sciences and Technologies*, (3), 175–187.

2. STM32F429ZG. High-performance advanced line, ARM Cortex-M4 core with DSP and FPU, 1 Mbyte Flash, 180 MHz CPU, ART Accelerator, Chrom-ART Accelerator, FMC with SDRAM, TFT. https://www.st.com/en/microcontrollers-microprocessors/stm32f429zg.html.

3. STM32H743ZI. High-performance and DSP with DP-FPU, Arm Cortex-M7 MCU with 2MBytes of Flash memory, 1MB RAM, 480 MHz CPU, Art Accelerator, L1 cache, external memory interface, large set of peripherals. https://www.st.com/en/microcontrollers-microprocessors/stm32h743zi.html.

4. ARM® Cortex®-M7 Processor Technical Reference Manual. URL: https://developer.arm.com/documentation/ddi0489/f.

5. IAR Embedded Workbench for Arm. https://www.iar.com/ewarm.

6. MDK Microcontroller Development Kit. https://www2.keil.com/mdk5.

7. STM32CubeIDE. Integrated Development Environment for STM32. https://www.st.com/en/development-tools/stm32cubeide.html.

8. Gabbrielli, M., & Martini, S. (2010). *Programming Languages: Principles and Paradigms*. Springer.

9. Vasiliev, O. (2017). *Programuvannia na C++ v prykladakh a zadachakh [C++ Programming In Examples And Exercises: Textbook Manual]*. Lira-K.

10. Banahan, M., Brady, D., & Doran, M. *The C Book – Table of Contents.* https://publications.gbdirect.co.uk//c_book/the_c_book.pdf.

11. *Beej's Guide to C Programming*. https://beej.us/guide/bgc/pdf/bgc_usl_c_2.pdf.

12. *Object-Oriented Programming with ANSI-C*. https://www.cs.rit.edu/ ~ats/books/ooc.pdf.

13. *Mathworks: Products and Services*. https://www.mathworks.com.

14. *Embedded Coder: Generate C and C++ code optimized for embedded systems*. https://www.mathworks.com/products/embedded-coder.html.

15. Voytenko, V. P., & Khomenko M. A. (2012). Signalnyi processor v sisteme upravleniya kvazirezonansnym preobrazovatelem [Signal processor in the control system of the quasi-resonant boost converter]. *Tekhnichna elektrodynamika. – Technical electrodynamics*. *Spec. issue "Power electronics and energetic efficiency", 2*, 101-106.

16. Kuhail M. A., Farooq S., Hammad R., & Bahja M. (2021). Characterizing Visual Programming Approaches for End-User Developers: A Systematic Review. *IEEE Access, 9*, 14181–14202. https://ieeexplore.ieee.org/document/9320477.

17. STM32 embedded target for MATLAB and Simulink with PIL and external mode processing (RN0087). https://www.st.com/en/development-tools/stm32-mat-target.html#documentatio.

18. STM32Cube initialization code generator. https://www.st.com/en/development-tools/stm32cubemx.html.

19. Hong, Ye (2021). *Port the Generated ARM Cortex-M CRL Code from MATLAB to KEIL µVision IDE*. https://www.mathworks.com/matlabcentral/fileexchange/48809-port-the-generated-arm-cortex-m-crl-code-from-matlab-to-keil-vision-ide, MATLAB Central File Exchange.

20. Khomenko, M., Voytenko, V., Vagapov, Y. (2013). Neural Network based Optimal Control of a DC Motor Positioning System. *Int. J. Automation and Control, 7*(1/2), 83-104.

Отримано 03.12.2021

UDC 004.453[629.735+004.896]

Volodymyr Voytenko[1], Roman Yershov[2]

[1]PhD in Technical Sciences, Associate Professor of the Electronics, Automation, Robotics and Mechatronics Department
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: volodymyr.voytenko@inel.stu.cn.ua. **ORCID:** http://orcid.org/0000-0003-1490-0600
ResearcherID: F-8698-2014. **Scopus Author ID**: 36167678700

[2]Senior Lecturer of the Electronics, Automation, Robotics and Mechatronics Department
Chernihiv Polytechnic National University (Chernihiv, Ukraine)
E-mail: roman.d.yershov@gmail.com. **ORCID**: https://orcid.org/0000-0002-0267-2906
ResearcherID: H-1432-2016. **Scopus Author ID:** 57188719994

ULTRA-HIGH-LEVEL PROGRAMMING OF THE SYSTEM OF ELECTRIC DRIVES OF QUADCOPTERS AND AUTONOMOUS ROBOTS

The research is devoted to the urgent task of reducing the time of software prototype development for the system of electric drives of unmanned aerial vehicles (UAVs) or autonomous robots (AR).

The development of software for the control system of UAVs and ARs requires in-depth knowledge of the problem area and practical skills of programming a specific microcontroller (MC), and the creation of power systems with new energy saving algorithms requires changing target platforms, which leads to additional training, development time and cost.

The most adequate component for the creation of control systems for UAV and AR is the ARM Cortex family MC. Integrated development environments (IDE) use high- or medium-level programming languages, which increases the amount and complexity of software code.

One of the stages in the development of a complex electronic system is to create a simulation model, most often in the MATLAB®-Simulink® environment. Thanks to the available means of generating software code for the development of a prototype of the control system of electric drives of a stand-alone device, you can use ultra-high-level languages.

The aim of the work is to analyze the existing possibilities of using ultra-high level programming languages in the computer modeling environment to create a software project of a prototype of UAV and AR control system, which allows to significantly reduce design time.

The possibilities of using MATLAB Coder™, Simulink Coder™ and Embedded Coder® are analyzed and the STM32 embedded target tool for MATLAB® and Simulink® is used to automate the process of generating the code of the built-in MC.

The efficiency of using the available ultra-high level programming tools for the development of control systems for UAV and AR electric drives on the basis of built-in MCs has been experimentally confirmed. Direct porting of automatically generated code to the IDE, as well as research into the use of the PIL (Processor-In-the-Loop) method, remains relevant.

Keywords: *electric drive model; MATLAB; Simulink; unmanned aerial vehicle (UAV); autonomous robot; software for the electric power supply.*

Fig.: 2. References: 20.