УДК 681.326

# ОГЛЯД ГРАФІЧНИХ БІБЛІОТЕК ДЛЯ ВБУДОВАНИХ ПЛАТФОРМ

*ФІЛІППЕНКО І.В., КОРНІЄНКО В.Р., КУЛАК Г.К.*

Наводиться огляд існуючих на поточний момент графічних бібліотек для вбудованих систем для створення панелей HMI та приладів носимої електроніки. Розглядаються основні етапи, які необхідні для адаптації однієї з існуючих бібліотек на процесор Nordic. Наведено вимоги до ресурсів системи (ОЗП, ПЗП, різновидам інтерфейсів, що наявні на конкретній платформі) для проектування вбудованих систем з використанням графічних пристроїв. Вивчаються особливості SDK для одного з процесорних вендорів та рішень, що необхідно реалізувати для запуску бібліотеки.

**Ключові слова:** графічна бібліотека, вбудована система, інтерфейси обміну даними, способи організації обміну, процесорний вендор, дисплейні модулі.

**Key words:** graphical library, embedded system, data transfer interface, data transferring organization, processors vendor, display modules.

## 1. Вступ

Одним з етапів проектування вбудованої системи є створення графічного інтерфейсу. Особливо важливим це є для області проектування HMI-пристроїв (Human-Machine Interface). З розповсюдженням на ринку 32-бітних процесорів на базі ядра ARM зростає тенденція до додавання графічного інтерфейсу та створення його близьким за виглядом до вже існуючих стандартів дизайну Material.

Змінюються тенденції в області підбору цільового мікроконтролера, який буде використано у пристрої. Поступово до периферії у процесорних вендорів, таких як STMicroelectronics, Nordic Semiconductor, Texas Instruments, Infineon, додаються просунуті модулі роботи з пам’яттю, графічні прискорювачі.

З метою забезпечення тривалої автономної роботи системи (зазвичай це стосується пристроїв носимої електроніки) використовують просунуті режими енергоспоживання та його профілювання. У випадку наявності графічної складової у системі потрібно забезпечити коректність інформації, що доступна на дисплеї під час роботи у режимі зниженого енергоспоживання.

Беручи до уваги постійне вдосконалення та існуючий різновид дисплейних інтерфейсів, необхідно враховувати можливості бібліотеки з боку виведення зображення, тому що іноді потрібна якість зображення не може бути досягнута через особливості конкретного інтерфейсу.

## 2. Існуючі інтерфейси обміну з дисплейними модулями

За типом обміну між контролером та периферійним дисплейним модулем інтерфейси поділяють на послідовні та паралельні. У випадку інтерфейсів з високою швидкістю слід відмітити паралельні FSMC та LTDC [1]. Розглянемо деякі з них.

**SPI** (Serial Peripheral Interface) – один з найвідоміших і широко використовуваних на даний час інтерфейсів зв'язку між різними мікросхемами в електронних пристроях. Його основними перевагами є простота, невелика кількість сигнальних ліній (у багатьох випадках достатньо лише трьох ліній), достатня простота реалізації друкованої плати при розробці пристрою.

Ключовою відмінністю SPI від паралельних інтерфейсів є те, що він має всього одну лінію передачі даних, та швидкість передачі часто не перевищує 20 Мбіт/с, що значно обмежує пропускну здатність, а отже і максимальний розмір керованого дисплея. Це робить практично неможливим створення складних динамічних зображень. Крім того, управління дисплеєм відбувається за допомогою постійного пересилання команд, а це змушує ядро контролера виконувати безліч зайвих задач, що підвищує його завантаження і робота з зображенням виявляється неефективною. У процесорів від вендора Espressif (сімейство ESP32) наразі доступні процесори з модулями SPI, що працюють на частоті до 60 МГц. У випадку старших сімейств STM32H7 доступна робота з шинами SPI на швидкості до 120МГц. Слід зауважити, що дисплейні контролери зазвичай підтримують роботу з шиною SPI на частоті не вище 50МГц.

**FSMC** (Flexible Static Memory Controller) – контролер зовнішньої пам'яті, який дозволяє взаємодіяти з TFT-дисплеями, забезпеченими інтерфейсами Motorola 6800 і Intel 8080. При певних умовах FSMC може використовуватися для роботи з TFT-дисплеями з RGB-інтерфейсом.

**LTDC** (LCD-TFT Display Controller) – спеціалізований TFT-контролер, який дозволяє працювати з TFT-дисплеями з RGB-інтерфейсом.

Послідовний інтерфейс, що набуває популярності у процесорів старших сімейств з ядром ARM, – **MIPI DSI** (Display Serial Interface специфікація Mobile Industry Processor Interface), який є спеціалізованим інтерфейсом для роботи з TFT-дисплеями, з MIPI-DSI [2].

Ці інтерфейси доступні лише на останніх сімействах процесорів, таких як STM32L4, STM32H7, STM32F7. На більш розповсюджених моделях зазвичай є у наявності інтерфейси SPI та I2C, в окремих випадках – інтерфейс LTDC.

У більшості випадків внутрішньої пам'яті мікроконтролера не вистачає для зберігання графічних зображень і розміщення екранної пам'яті. З цієї причини дуже часто потрібна зовнішня Flash-пам’ять, або ОЗП. Для підключення зовнішньої

Flash-пам’яті доцільно використовувати інтерфейс QuadSPI, який присутній практично у всіх останніх лінійках STM32. Для підключення до STM32 додаткового ОЗП слід використовувати контролер зовнішньої пам'яті FSMC / FMC. Головна відмінність між FSMC і FMC полягає в тому, що FMC здатний працювати з SDRAM. До складу багатьох мікроконтролерів STM32 входять і інші допоміжні блоки, створені для оптимізації роботи з графікою. Вони не є обов'язковими для створення графічних додатків, але їх використання значно скорочує навантаження на процесорне ядро. Це такі блоки, як:

Chrom-ART – графічний прискорювач. Являє собою особливий контролер прямого доступу до пам'яті, створений спеціально для оптимізації процесів з пересилання великих масивів графічних даних з можливістю їх потокової обробки;

Chrom-GRC – блок оптимізації зберігання графічних даних, що дозволяє економити до 20% пам'яті при роботі з дисплеями округлої форми;

JPEG-кодек – спеціалізований прискорювач, необхідний для JPEG-кодування і декодування графічних зображень.

**3. Огляд існуючих пропріетарних бібліотек з закритим кодом**

Існує декілька різновидів графічних бібліотек для вбудованих систем, які розрізняються за системою ліцензування.

Ринок пропріетарних бібліотек є досить розвиненим за рахунок кросс-платформенних рішень у цій області. На даний момент користується високою популярністю розробка Qt for MCU, що дозволяє використовувати майже всі потужності Qt Framework для програмування під десктопні платформи для запуску додатків на базі вбудованих платформ.

Можливості Qt Framework для мікроконтролерів є такі [3]:

– підтримка Qt Quick для швидкого прототипування графічних інтерфейсів з використанням технології QML та мов програмування C/C++ для написання бізнес-логіки додатку;

– підтримка Qt Design Studio для формування графічного інтерфейсу та його перегляду без використання цільової платформи;

– підтримка Qt Creator – інтегрованого середовища розробки кросс-платформенних додатків, розробки додатків для платформ Anroid та iOS. Додатково є підтримка технології WebAssembly для створення додатків для Web (для додатків зі спільною кодовою базою).

На сайті проекту доступні прошивки з демонстраційним програмним забезпеченням та демонстрації графіки на базі Qt для серій процесорів з вбудованими графічними прискорювачами, такими як PxP від виробника NXP для серії i.MX RT, Chrom-Art Accelerator на старших сімействах процесорів STM32 та RGL на процесорах Renesas RH580. Бібліотека дозволяє використовувати таку зв’язку мов програмування: прототипування інтерфейсу на базі QML + JS та розробка бізнес-логіки додатку з використання мов програмування C/C++.

EmbeddedWizard – розробка компанії Tara Systems, спрямована на спрощення створення графічного інтерфейсу. Споріднює у собі використання внутрішнього середовища розробки Embedded Wizard Studio [4]. Особливість середовища – підтримка WYSIWIG розробки інтерфейсу у поєднанні з генератором коду для цільової платформи. Використання цього підходу дозволяє швидко розробляти графічний інтерфейс без необхідності огляду на апаратно-залежні компоненти.

Архітектура бібліотеки заснована на розділі платформо-залежних компонентів та провадження генераторів коду під цільову платформу. Типове використання бібліотеки полягає у створенні прототипу у Embedded Wizard Studio, конвертації ресурсів та генерації коду під конкретну платформу (рис. 1).

До недоліків запропонованої бібліотеки слід віднести недостатність документації щодо створення апаратно-залежних компонентів та їх використання [5].

Основна мова програмування – Chora, яка була побудована за мотивами C++ та Java для створення максимально подібного синтаксису. Chora підтримує концепцію GC (Garbage Collector) для менеджменту ресурсів системи та об’єктів в ній. Наявна підтримка моделі сигналів-слотів для взаємодії об’єктів графічного інтерфейсу та бізнес-логіки.

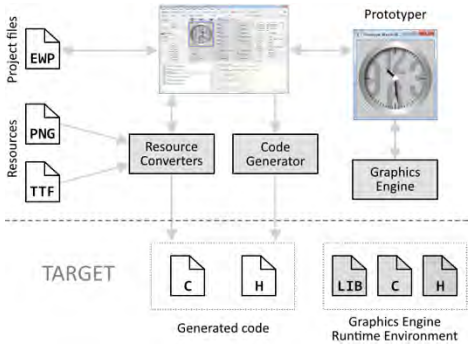


Рис. 1. Загальна інфраструктура бібліотеки Embedded Wizard

EasyGui – розробка компанії IBIS Solutions (рис. 2). Проект не має розвитку з 2015 року. На поточний момент має базову підтримку віджетів, підтримку як кольорових, так і монохромних дисплеїв [6]. Розробка інтерфейсу проходить з

використанням програми easyGUI PC Application, що надає доступ до таких компонентів:
easyGui data files – файли шрифтів, зображень, згенерованого графічного інтерфейсу. При кожній зміні будь-чого у програмі-дизайнері ці файли мають бути перегенеровані;
easyGui library – безпосередньо реалізація бібліотеки;
easyGui display driver – платформо-залежний драйвер дисплею, що буде використаний у проекті; easyGui має у комплекті поставки набір драйверів, що можуть бути використані.

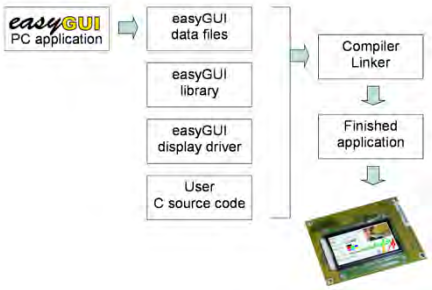


Рис. 2. Генерація проекту при використанні бібліотеки easyGUI

TouchGFX на даний час є власністю компанії STMicroelectronics та пропонує набір рішень в області швидкого прототипування графічного інтерфейсу. Проект має можливість швидко інтегруватися у існуючі системи розробки на базі STM32. Інтеграція відбувається як модуль до пакету STM32Cube. TouchGFX поставляється як модуль X-CUBE-TOUCHGFX. Проект просувається як рішення для створення панелей HMI.
Серед особливостей проекту – просунуте середовище прототипування, а саме: середовище розробника TouchGFX Designer з можливістю швидкого завантаження отриманого інтерфейсу на цільову платформу [7].
Бібліотека підтримує наступні роботи з дисплейними буферами:
SingleBuffer – використовується буфер розміром з цільовий дисплей. Для оптимізації рисування може бути використаний функціонал, що дозволяє визначити ту частину області буфера, яка відрисована на дисплей, та помітити її як вільну;
Double Buffer – подвійна буферизація;
Partial buffer – дозволяє використовувати необхідну кількість блоків, що задаються користувачем. Для ініціалізації необхідно задати кількість блоків, що використовуються, та розмір одного блока. Приклад наведено у Лістингу 1.
Приклади використання бібліотеки у різних режимах передачі зображення на дисплей наведені у [8], а саме приклади передачі даних з використанням інтерфейсів MIPI DSI для плати на базі STM32L4R9 та SPI для плати налагодження на базі STM32G081.
Використання бібліотеки можливо як з наявністю операційної системи реального часу, так і без її використання. Бібліотека потребує періодичного виклику функції void MX_TouchGFX_Process(void); для обробки циклу повідомлень.

Лістинг 1
Приклад встановлення стратегії використання дисплейного буфера кадрів у графічній бібліотеці Touch GFX

```
/2 або більше блоків розміром 10*390 пікселів
ManyBlockAllocator<10*390*3, 2, 3>
frameBufferAllocator;
void touchgfx_init()
{
  HAL& hal = touchgfx_generic_init(dma, display, tc,
GUI_DISPLAY_WIDTH,
                    GUI_DISPLAY_HEIGHT, 0, 0, 0);
  hal.setFrameBufferStartAddress((uint16_t*)0,
GUI_DISPLAY_BPP, false, false);
  hal.setFrameBufferAllocator(&frameBufferAllocator);

hal.setFrameRefreshStrategy(HAL::REFRESH_STRATE
GY_PARTIAL_FRAMEBUFFER);
}
```

## 4. Огляд існуючих бібліотек з відкритим кодом

EmWin є розробкою компанії Segger, яка на даний час є одним з найбільших виробників обладнання для налагодження та проектування пристроїв на базі процесорів архітектури ARM. Бібліотека є частково платною – за комерційною ліцензією надається доступ до додаткових модулів. В концепцію бібліотеки покладена ідея, схожа на механізм роботи віконної підсистеми операційної системи Windows.
У деяких випадках схожість є навіть у типах повідомлень, структурі та механізмах обробки.
В основу проектування покладено базовий примітив – вікно. У додаткових програмних модулях, що доступні за комерційною ліцензією, слід відмітити компоненти emVNC клієнт, emWinSPY та emWinView [9] (рис. 3, 4).
Компонент emVNC дозволяє розташувати на цільовому пристрої клієнта Virtual Network Computing, що дозволяє виконувати дистанційне керування пристроєм, передачу файлів та дистанційне налагодження пристрою.
EmWinSpy дозволяє виконувати профілювання споживання оперативної пам’яті на цільовому пристрої, спостерігати за контентом, що відображується на користувацькому дисплеї, та робити скріншоти з цільового пристрою. Можливо також виконувати логування даних користувацького введення, наприклад, з клавіатури, тачскріна або підтримку MultiTouch введення.

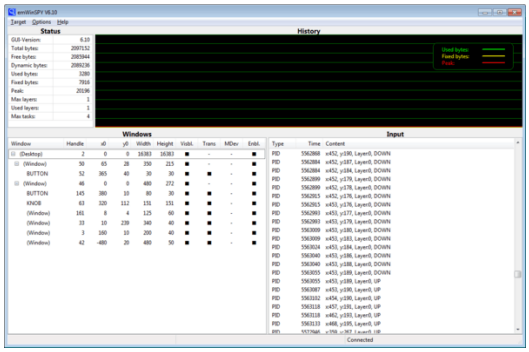

Рис. 3. Середовище налагодження emWinSpy

Компонент emWinView дозволяє переглядати контент дисплею в окремому процесі при використанні симулятора для десктопних платформ [10]. Зазвичай при паузі потоку, що знаходиться у режимі налагодження, виконується зупинка всіх потоків, які належать до запущеного процесу. EmWinView вирішує цю проблему шляхом переносу відображення контенту дисплею в окремий процес. Утиліта може бути запущена як до, так і під час налагодження.

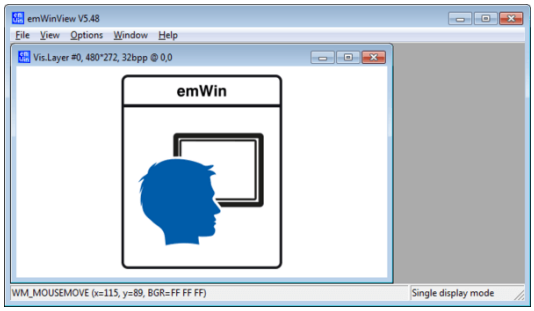


Рис. 4. Утиліта emWinView

LittlevGL – бібліотека з відкритим програмним кодом, що зараз набуває популярності завдяки програмним рішенням та особливостям. Вона є актуальною для малих вбудованих систем [11]. Як і TouchGFX, бібліотека підтримує три стратегії роботи з відеобуферами для виводу інформації на дисплей. Головними особливостями бібліотеки є гнучкість конфігурування та можливість роботи на декількох дисплеях одночасно, що може бути необхідним для імплементації TileView на вбудованій платформі.

Опції конфігурування у бібліотеці дозволяють гнучко сконфігурувати необхідний функціонал, відключаючи непотрібні опції для зображення та віджети, що не будуть використані у ході роботи з бібліотекою. Бібліотека підтримує можливість змінювати зовнішній вигляд компонентів шляхом використання вбудованих тем або створення користувацьких. Є можливість інтеграції з Touch-Screen контролерами, енкодерами та користувацькими кнопками. Інтеграція проходить шляком реалізації інтерфейса lv_indev_drv_t. Наприклад, для ініціалізації підтримки енкодера необхідно реалізувати функцію, що буде викликана у lv_indev_drv_t.read_cb, та встановити тип пристрою - джерело подій. Всього у бібліотеці на момент версії 6.0 існує чотири джерела подій, а саме:

LV_INDEV_TYPE_POINTER – контролер TouchScreen або миші;

LV_INDEV_TYPE_KEYPAD – клавіатура або кейпад;

LV_INDEV_TYPE_ENCODER – енкодер з опціональною кнопкою;

LV_INDEV_TYPE_BUTTON – зовнішня користувацька кнопка, що може бути прив'язана до конкретного віджету.

Приклад реалізації роботи з енкодером наведено у Лістингу 2.

Лістинг 2
Приклад обробки енкодера у LittlevGL

```
indev_drv.type = LV_INDEV_TYPE_ENCODER;
indev_drv.read_cb = my_input_read;

...

bool encoder_read(lv_indev_drv_t * drv,
lv_indev_data_t*data){
  data->enc_diff = enc_get_new_moves();

  if(enc_pressed()) data->state =
LV_INDEV_STATE_PR;
  else data->state = LV_INDEV_STATE_REL;

  return false; /*No buffering now so no more data read*/
}
```

Бібліотека підтримує роботу з задачами, а саме можливість періодичного або однократного виклику.

## 5. Фрагменти адаптації бібліотеки Little VGL для плати на базі процесора NRF52832

Як приклад реалізації адаптації відкритої бібліотеки LVGL було розроблено стенд на базі модуля E73NRF52832 з популярним дисплеєм на базі контролера Sitronix. Для реалізації дисплейного драйвера було обрано мову програмування C++17, з боку SDK від Nordic було обрано використання версії 15.3. На даний момент актуальною є версія SDK-16.

З вимог до бібліотеки LVGL необхідно виконати такі дії. Спочатку обрати тип буфера, що буде використаний, далі провести ініціалізацію драйвера дисплею та призначити функцію, що реалізує заповнення області дисплею зазначеного розміру зазначеним кольором. Опціонально реалізувати заповнення за допомогою GPU, режими змішування кольорів на GPU метод округлення розмірів області, що потребує оновлення

(для дисплеїв з особливими режимами роботи з зонами рисування). Опціонально реалізувати метод виведення пікселя на дисплей у задані координати. Для профілювання виведення на дисплей додати monitor-метод, у якому виконувати виведення часу оновлення зони на дисплеї та кількість пікселів, що була оновлена.

З боку драйвера для дисплею потрібно реалізувати відправлення команд управління та даних на дисплей. У випадку відправлення команди потрібно встановлювати порт керування D/C (Data/Command) у необхідний стан залежно від типу даних, що будуть передані [12].

Для розвантаження процесора та досягнення більшої ефективності при передачі даних слід використати модуль EasyDMA у процесора Nordic.

EasyDMA – це простий у використанні модуль прямого доступу до пам'яті, який реалізує взаємозв'язок між периферійними пристроями та оперативною пам'яттю даних. EasyDMA є майстром шини AHB, подібно ядру контролера він підключений до багатошарового з'єднання AHB. Однак при цьому EasyDMA не може отримати доступ до Flash [13].

Периферійний пристрій може надати декілька каналів за допомогою EasyDMA одночасно, наприклад, при забезпеченні виділеного каналу для зчитування даних з ОЗП в периферійний пристрій одночасно, коли другий канал призначений для запису даних в ОЗП з периферії. Ця концепція проілюстрована на рис. 5.

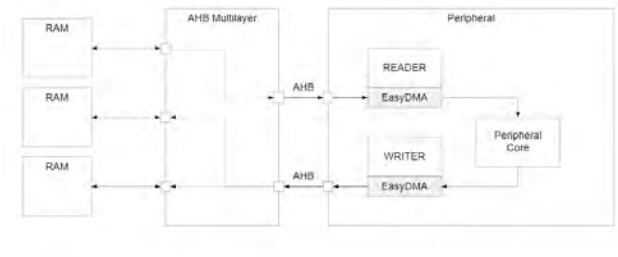


Рис. 5. Структурна схема EasyDMA

SDK-сигнатура передачі даних шиною SPI за допомогою EasyDMA має вигляд:

```
nrfx_err_t nrfx_spim_xfer(nrfx_spim_t const * const p_instance, nrfx_spim_xfer_desc_t const * p_xfer_desc, uint32_t flags);
```

де структура дескриптора передачі виглядає так:

```
typedef struct
{
    uint8_t const * p_tx_buffer; ///< Pointer to TX buffer.
    size_t          tx_length;   ///< TX buffer length.
    uint8_t       * p_rx_buffer; ///< Pointer to RX buffer.
    size_t          rx_length;   ///< RX buffer length.
} nrfx_spim_xfer_desc_t;
```

Залежно від структури дескриптора, розмір блока передачі даних відповідає типу size_t, що на 32 платформі дорівнює 2^32 елементам. У ході роботи з логічним аналізатором було визначено, що за одну транзакцію на шині передача більше, ніж 255 байт, неможлива. З використанням тестового прикладу передачі даних була виконана передача транзакції розміром 255 байтів. Запис з логічного аналізатора наведено на рис. 6, після чого розмір транзакції було розширено до 257 байтів.

Отриманий запис передачі наведено на рис. 7. З цього можна зробити висновок про необхідність реалізації транзакційної системи передачі даних за шиною SPI [14]. Для цього можна використовувати або SPI transaction manager, що входить у NordicSDK, або власну реалізацію на базі реалізації TrasnsactionItem.

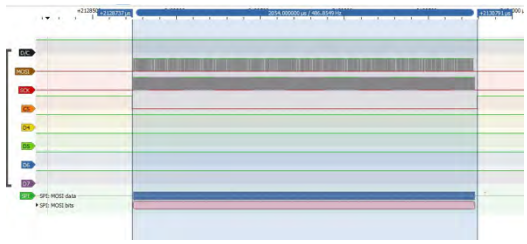

Рис. 6. Запис трафіка з шини SPI при виконанні транзакції

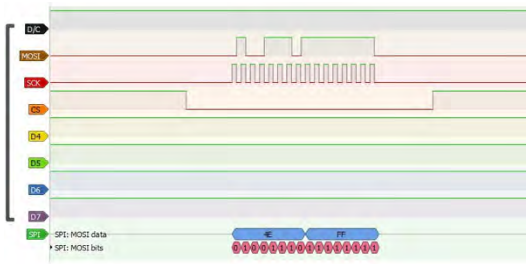


Рис. 7. Запис неповної передачі буфера при невірному використанні NordicSDK

Для портування бібліотеки необхідно з драйвера дисплейного модуля надати метод fillRectangle, що заповнює прямокутник заданого розміру заданим буфером кольору. Після повної передачі буфера потрібно викликати lv_disp_flush_ready. Як приклад було реалізовано передачу буфера фрагментами по 255 байтів (щоб забезпечити повну транзакцію) та неповним буфером. Якщо розмір не вкладається у 255 байтів, для цього у компоненті драйвера розроблена логіка на базі компонента TransactionItem [15].

При цьому необхідно врахувати розмір буфера у байтах, що буде переданий. Необхідно також розрахувати кількість повних транзакцій та кількість неповних блоків. У Лістингу 3 наведено фрагмент реалізації передачі повного блока даних на дисплей.

Лістинг 3

Фрагмент реалізації повної передачі даних на дисплей

```
    if( FullDmaTransactionsCount > 0 )
  {
    Interface::Spi::Transaction fullTransaction{};
    fullTransaction.beforeTransaction =
      [ this ]
      {
        setDcPin();
      };

    fullTransaction.transactionAction =
      [this, _colorToFill]
      {
        std::uint32_t addrOffset =
Interface::Spi::SpiBus::DmaBufferSize
                * getTransitionOffset();

        m_pBusPtr->sendChunk(
              reinterpret_cast<const std::uint8_t*>(
_colorToFill ) + addrOffset
            , Interface::Spi::SpiBus::DmaBufferSize
        );
      };

    if( FullDmaTransactionsCount > 1 )
      fullTransaction.repeatsCount                    =
FullDmaTransactionsCount - 1;

    if( ChunkedTransactionsCount == 0 )
      fullTransaction.afterTransaction =
        [this]
        {
          onRectArreaFilled.emit();
          resetDcPin();
        };

    m_pBusPtr->addTransaction( std::move(
fullTransaction ) );
  }
```

Результат запуску бібліотеки LittlevGL наведено на рис.8:

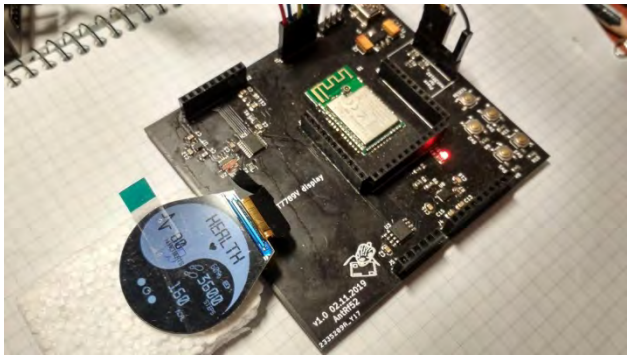

Рис. 8. Спроектований графічний інтерфейс з використанням бібліотеки LittlevGL

## 6. Висновки

Розглянуто бібліотеки, які використовуються для роботи з графічними пристроями. Розглянуто бібліотеку TouchGFX, яка є зручною у використанні в області Wearable devices. Показано, що інструменти на базі Qt Framework використовуються для побудови рішень в області Automotive solutions. LittlevGL project є альтернативою комерційним компонентам, що надає доступний та необхідний функціонал під ліцензією розповсюдження MIT. Інші розглянуті бібліотеки (emWin, easyGUI, Embedded Wizard) є активними у використанні HMI та прототипуванні вбудованих пристроїв.

**Література:** **1.** *Компел.* Розробляємо пристрій з графікою на STM32. Основні можливості периферії [Електронний ресурс] / Компел – Режим доступу: https://www.compel.ru/lib/90626.. 2018. **2.** *Компел.* STM32 + сучасний TFT-дисплей: варіанти на будь-який смак [Електронний ресурс] / Компел – Режим доступу: https://www.compel.ru/lib/88474. **3.** *Holländer P.* Announcing Qt for MCUs - A comprehensive toolkit [Електронний ресурс] / Petteri Holländer – Режим доступу: https://www.qt.io/blog/2019/08/21/announcing-qt-mcus. **4.** *Embedded Wizard.* Simplify Your GUI DevelopmentSimplify Your GUI Development [Електронний ресурс] / Embedded Wizard – Режим доступу до ресурсу: https://www.embedded-wizard.de/. **5.** *GitHub.* Microgui Description [Електронний ресурс] / GitHub – Режим доступу: https://github.com/ryankurte/micro-gui. **6.** *EasyGUI.* easyGUI information downloads [Електронний ресурс] / easyGUI – Режим доступу: https://www.easygui.com/easygui-downloads. **7.** *TouchGFX.* The TouchGFX Abstraction Layer (AL) [Електронний ресурс] / TouchGFX – Режим доступу: https://support.touchgfx.com/docs/development/touchgfx-hal-development/touchgfx-al-development-Introduction. **8.** *TouchGFX.* Lowering Memory Usage with Partial Framebuffer [Електронний ресурс] / TouchGFX –: https://support.touchgfx.com/docs/development/ui-development/scenarios/lowering-memory-usage-with-partial-framebuffer/. **9.** *SEGGER. emWin* – The professional GUI for embedded devices enables the creation of highly efficient and high quality graphical [Електронний ресурс] / SEGGER – Режим доступу: https://www.segger.com/products/user-interface/emwin/. **10**. EmWinView [Електронний ресурс] // SEGGER. 2019. https://www.segger.com/products/user-interface/emwin/tools/tools-overview/. **11.** *LittlevGL.* Open-source Embedded GUI Library [Електронний ресурс] / LittlevGL – https://littlevgl.com/. **12.** *Джозеф Ю.* Ядро Cortex-M3 компанії ARM. Повне керівництво / Ю. Джозеф., 2012. 552 с. **13.** *Nordic Semiconductor Infocenter*. EasyDMA [Електронний ресурс] / Nordic Semiconductor Infocenter – Режим доступу до ресурсу: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%2Feasydma.html. **14.** *Kormanyos C.* Real-Time C++: Efficient Object-Oriented and Template Microcontroller Programming / Christopher Kormanyos., 2018. 426 с. **15.** *Arobenko A.* Practical Guide to Bare Metal C++ [Електронний

ресурс] / Alexander Arobenko – Режим доступу: https://arobenko.gitbooks.io/bare_metal_cpp/content/.

**Transliterated bibliography:**

**1.** *Kompel.* Rozrobljaємo pristrij z grafikoju na STM32. Osnovni mozhlivosti periferiï [Elektronnij resurs] / Kompel Rezhim dostupu do resursu: https://www.compel.ru/lib/90626.. 2018

**2.** *Kompel*. STM32 + suchasnij TFT-displej: varianti na bud'-jakij smak [Elektronnij resurs] / Kompel – Rezhim dostupu do resursu: https://www.compel.ru/lib/88474.

**3.** Holländer P. Announcing Qt for MCUs - A comprehensive toolkit [Elektronnij resurs] / Petteri Holländer – Rezhim dostupu do resursu: https://www.qt.io/blog/2019/08/21/announcing-qt-mcus.

**4.** *Embedded* Wizard. Simplify Your GUI DevelopmentSimplify Your GUI Development [Elektronnij resurs] / Embedded Wizard – Rezhim dostupu do resursu: https://www.embedded-wizard.de/.

**5.** *GitHub*. Microgui Description [Elektronnij resurs] / GitHub – Rezhim dostupu do resursu: https://github.com/ryankurte/micro-gui.

**6.** *EasyGUI.* easyGUI information downloads [Elektronnij resurs] / easyGUI – Rezhim dostupu do resursu: https://www.easygui.com/easygui-downloads.

**7.** *TouchGFX.* The TouchGFX Abstraction Layer (AL) [Elektronnij resurs] / TouchGFX – Rezhim dostupu do resursu: https://support.touchgfx.com/docs/development/touchgfx-hal-development/touchgfx-al-development-Introduction.

**8.** *TouchGFX*. Lowering Memory Usage with Partial Framebuffer [Elektronnij resurs] / TouchGFX –: https://support.touchgfx.com/docs/development/ui-development/scenarios/lowering-memory-usage-with-partial-framebuffer/.

**9.** *SEGGER*. emWin – The professional GUI for embedded devices enables the creation of highly efficient and high quality graphical [Elektronnij resurs] / SEGGER – https://www.segger.com/products/user-interface/emwin/.

10 emWinView [Elektronnij resurs] // SEGGER. 2019. https://www.segger.com/products/user-interface/emwin/tools/tools-overview/.

**11.** *LittlevGL*. Open-source Embedded GUI Library [Elektronnij resurs] / LittlevGL – https://littlevgl.com/.

**12.** *Dzhozef Ju.* Jadro Cortex-M3 kompanії ARM. Povne kerivnictvo / Ju. Dzhozef., 2012. 552 s.

**13.** *Nordic* Semiconductor Infocenter. EasyDMA [Elektronnij resurs] / Nordic Semiconductor Infocenter – https://infocenter.nordicsemi.com/index.jsp?topic=%2Fcom.nordic.infocenter.nrf52832.ps.v1.1%2Feasydma.html.

**14** *Kormanyos C*. Real-Time C++: Efficient Object-Oriented and Template Microcontroller Programming / Christopher Kormanyos. 2018. 426 s.

**15.** *Arobenko A.* Practical Guide to Bare Metal C++ [Elektronnij resurs] / Alexander Arobenko – https://arobenko.gitbooks.io/bare_metal_cpp/content/.

Поступила в редколлегию 29.11.2019

**Рецензент**: д-р техн. наук, проф. Кривуля Г.Ф.

**Філіппенко Інна Вікторівна**, канд. техн. наук, доцент кафедри АПОТ ХНУРЕ. Наукові інтереси: вбудовані системи, цифрові фільтри. Адреса: Україна, 61166, Харків, пр. Науки, 14, тел. 702-13-26.

**Корнієнко Валентин Русланович**, студент ХНУРЕ. Наукові інтереси: HMI-системи, вбудовані системи, Embedded C++. Адреса: Україна, 61166, Харків, пр. Науки, 14, тел. 702-13-26, e-mail: valentyn.korniienko1@nure.ua.

**Кулак Георгій Костянтинович,** студент ХНУРЕ. Наукові інтереси: вбудовані системи, цифрові автомати. Адреса: Україна, 61166, Харків, пр. Науки, 14, тел. 702-13-26, e-mail heorhii.kulak@nure.ua.

**Filippenko Inna Victorovna**, PhD, Associate Professor, Design Automation Department, NURE. Scientific interests: embedded systems, digital filters. Address: Ukraine, 61166, Kharkiv, Nauky Ave. 14, tel. 702-13-26.

**Korniienko Valentyn Ruslanovich**, student, NURE. Scientific interests: HMI systems, embedded systems, Embedded C++. Address: Ukraine, 61166, Kharkiv, Nauky Ave. 14, tel. 702-13-26, e-mail: valentyn.korniienko1@nure.ua.

**Kulak Georgiy Konstantinovich**, student, NURE. Scientific interests: HMI systems, embedded systems, finite state machine. Address: Ukraine, 61166, Kharkiv, Nauky Ave. 14, tel. 702-13-26, e-mail: heorhii.kulak@nure.ua.