https://doi.org/10.31891/2219-9365-2024-80-10
УДК 621.382

ОСАДЧУК Олександр
Вінницький національний технічний університет
https://orcid.org/0000-0001-6662-9141
e-mail: osadchuk.av69@gmail.com
ОСАДЧУК Ярослав
Вінницький національний технічний університет
https://orcid.org/0000-0002-5472-0797
e-mail: osadchuk.j93@gmail.com
СКОЩУК Валентин
Вінницький національний технічний університет
e-mail: skoschuk999@gmail.com

ІНТЕГРАЦІЯ LILYGO LORA32 У БАГАТОКАНАЛЬНУ РАДІОТЕХНІЧНУ СИСТЕМУ НА FPGA ДЛЯ ЧАСТОТНИХ ПЕРЕТВОРЮВАЧІВ ФІЗИЧНИХ ВЕЛИЧИН

У роботі представлено розширення функціональних можливостей багатоканальної системи паралельного вимірювання частоти на основі FPGA фірми Altera Cyclone IV. Основним завданням є інтеграція модуля Lilygo LORA32, що забезпечує можливість попередньої обробки та передачі даних через бездротові канали зв'язку, такі як Lora, WiFi та Bluetooth. Удосконалено багатоканальний універсальний вимірювальний прилад, який має 12 вимірювальних каналів для сенсорів з частотним виходом, базується на мікропроцесорному ядрі NIOS II та може взаємодіяти з цифровими сенсорами фізичних величин. Описано налаштування середовища розробки, створення драйвера UART для зчитування виміряних даних, парсинг отриманих даних, обробку та перевірку помилок, а також запаковування даних у компактний формат для подальшої передачі. Проведено експериментальні дослідження та аналіз результатів, що демонструють ефективність і надійність запропонованої системи.

Ключові слова: LILYGO LoRa32, LoRa, WiFi, Bluetooth, NIOS II, FPGA, багатоканальний частотомір, сенсор з частотним виходом, радіовимірювальні перетворювачі фізичних величин, частота.

OSADCHUK Oleksandr, OSADCHUK Iaroslav, SKOSHCHUK Valentyn
Vinnytsia National Technical University

INTEGRATION OF LILYGO LORA32 INTO A MULTI-CHANNEL RADIO ENGINEERING SYSTEM ON FPGA FOR FREQUENCY CONVERTERS OF PHYSICAL QUANTITIES

The work presents the enhancement of the functional capabilities of a multi-channel parallel frequency measurement system based on the Altera Cyclone IV FPGA. The main task is the integration of the Lilygo LORA32 module, providing the capability for preliminary data processing and transmission through wireless communication channels such as Lora, WiFi, and Bluetooth. The multi-channel universal measuring device, which features 12 measurement channels for frequency output sensors, is based on the NIOS II microprocessor core and can interact with digital sensors of physical quantities. The paper describes the development environment setup, creation of a UART driver for reading measured data, parsing of received data, error handling and checking, and packaging data into a compact format for subsequent transmission. Experimental studies and analysis of the results demonstrate the efficiency and reliability of the proposed system. The Lilygo LORA32 module was selected for this project due to its numerous advantages and impressive characteristics that make it suitable for integration into advanced measurement systems. The Lilygo LORA32 features an ESP32 microcontroller, which provides robust processing power and support for various communication protocols, including Lora, WiFi, and Bluetooth. This versatility allows for flexible and efficient data transmission over long distances, ensuring reliable connectivity even in challenging environments. The key advantages of the Lilygo LORA32 include its low power consumption, which is critical for battery-operated devices, and its high sensitivity, which enhances signal reception and extends communication range. Additionally, the module supports multiple frequency bands, making it adaptable to different regional requirements and ensuring compliance with global communication standards. The integration of an OLED display on the module also allows for real-time data monitoring and debugging, further enhancing its usability in complex systems. Overall, the Lilygo LORA32's combination of processing power, communication versatility, and energy efficiency makes it an ideal choice for enhancing the capabilities of multi-channel measurement systems on FPGA platforms.

Keywords: LILYGO LoRa32, NIOS II, FPGA, multichannel frequency meter, sensor with frequency output, radio measuring transducers of physical quantities, frequency.

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Поява FPGA дала змогу всі елементи фізично розмістити на одній інтегральній схемі, та перейти на якісно новий рівень, який пов'язаний із значним збільшенням їх ступеня інтеграції і підвищенням швидкодії [1-4]. Поява NIOS II [5-9] дала змогу перейти на новий рівень абстракції при програмуванні FPGA, тепер не потрібно задумуватися про реалізацію: регістрів, арифметичних блоків, блоків пам'яті, шин передачі даних

і способу синхронізації вище згаданих блоків. На сучасному етапі розвитку радіотехнічних систем та вимірювальної техніки виникає необхідність у створенні багатофункціональних пристроїв, які можуть забезпечувати високу точність вимірювань та бездротову передачу даних. Існуючі системи часто обмежені можливостями проводового зв'язку, що ускладнює їх використання в умовах, де потрібна мобільність та оперативність отримання даних. Вимірювальні системи на основі FPGA зарекомендували себе як ефективні інструменти для паралельного вимірювання частоти, проте вони обмежені в можливостях бездротової передачі даних. Інтеграція сучасних бездротових модулів, таких як Lilygo LORA32, у вимірювальні системи на FPGA відкриває нові перспективи для їх використання, дозволяючи здійснювати передачу даних через канали зв'язку Lora, WiFi та Bluetooth.

АНАЛІЗ ОСТАННІХ ДОСЛІДЖЕНЬ

У роботі [2] наведено приклад розробки багатоканальної системи вимірювання частоти на FPGA. В якості мікросхеми для реалізації частотоміра використовується FPGA фірми Altera Cyclone IV EP4CE10F17C8. На рис. 1 зображено блок схема кінцевої системи. Основний її функціонал полягає у вимірюванні частоти з вхідних каналів, формування пакету з виміряними даними і надсилання даних через UART.

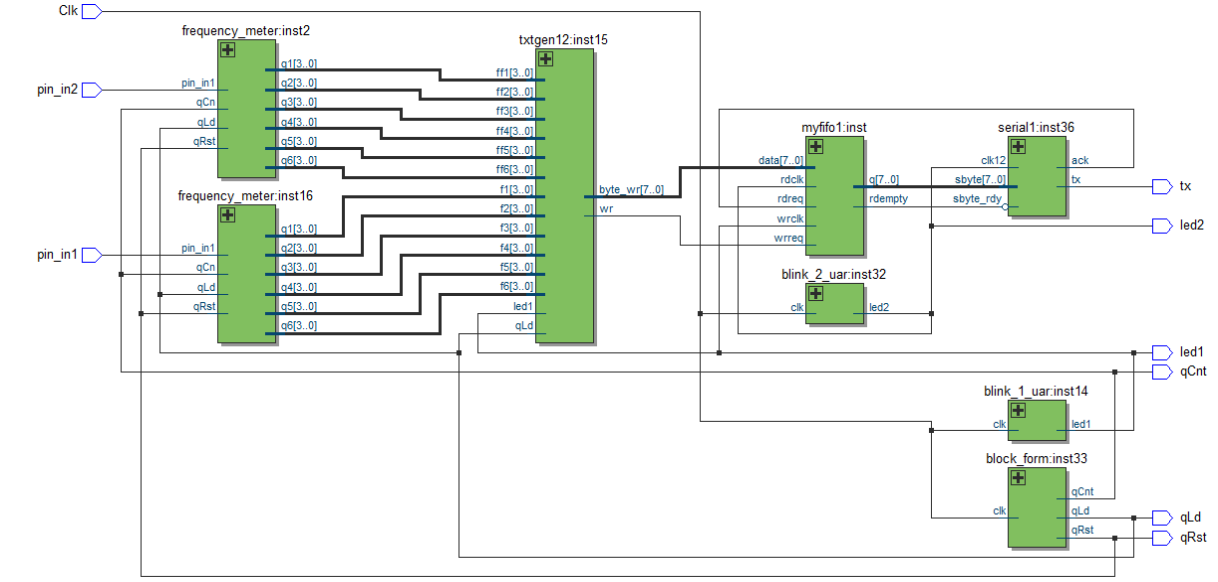


Рис. 1. Схема багатоканального частотоміра

У роботі [3] було запропоновано оптимізацію системи за рахунок інтеграції мікропроцесорного ядра NIOS II. Інтеграція NIOS II у багатоканальний частотомір дозволила зробити систему гнучкою, додати попередню обробку і фільтрацію отриманих даних на низькому рівні, змінювати кількість частотомірів, без зміни алгоритму обробки даних, що являлося неможливим у попередній реалізації [2]. На рис. 2 зображено блок схема кінцевої системи.

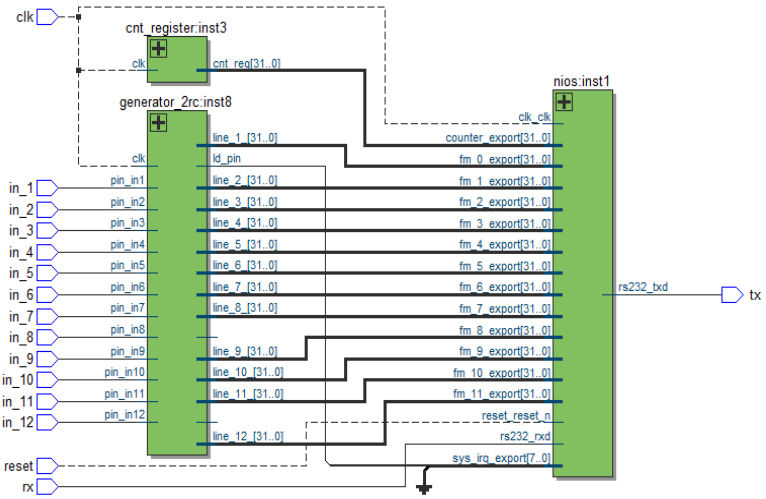


Рис. 2. Схема багатоканального частотоміра з використанням ядра NIOS II

У роботі [4] розширено функціонал системи [3] за рахунок інтеграції інтерфейсу для взаємодії із цифровими сенсорами - I2C. Для цього було створено інтерфейс ядра NIOS II для I2C, реалізовано I2C протокол у вигляді апаратного блоку, написано ПЗ для підтримки I2C шини. На рис. 3 зображено блок схему кінцевої системи.

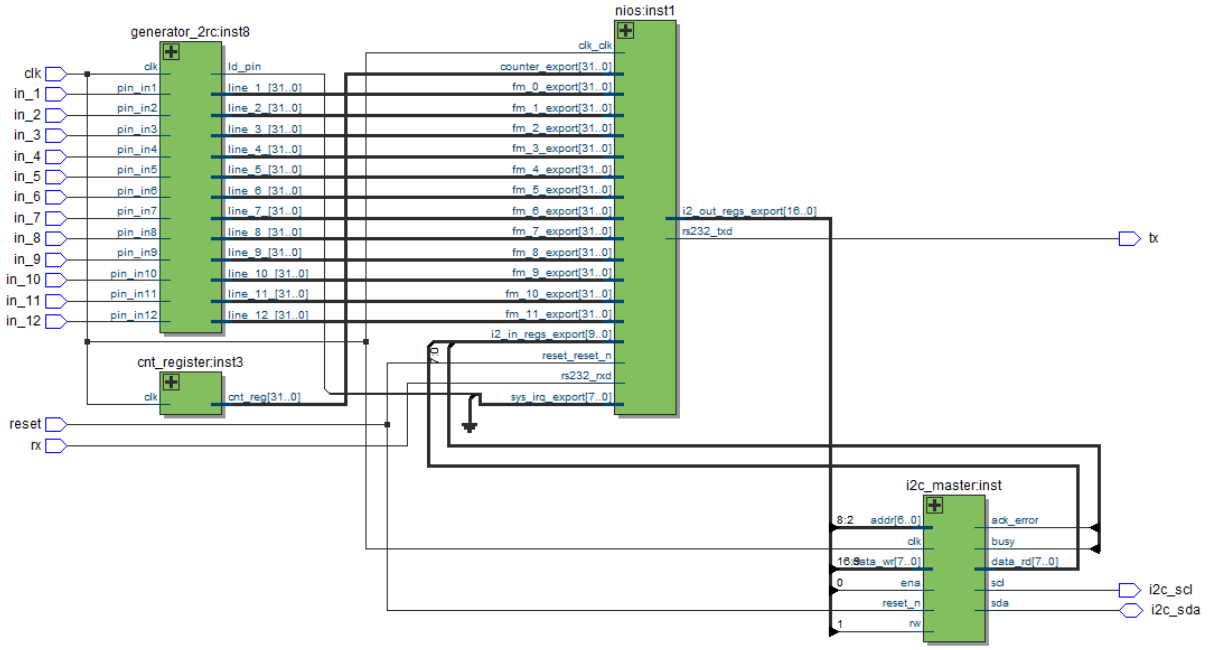


Рис. 3. Схема багатоканального частотоміра з використанням ядра NIOS II і підтримкою I2C протоколу

На рис. 4 зображено результат роботи розробленої системи при роботі із під'єднаним генератор частоти на 85 кГц і цифровими сенсорами. Перші 12 значень відповідають значенням частотомірів, наступне – значення температури, наступні три – значення акселерометра і останні три – значення гіроскопа, рис. 4.

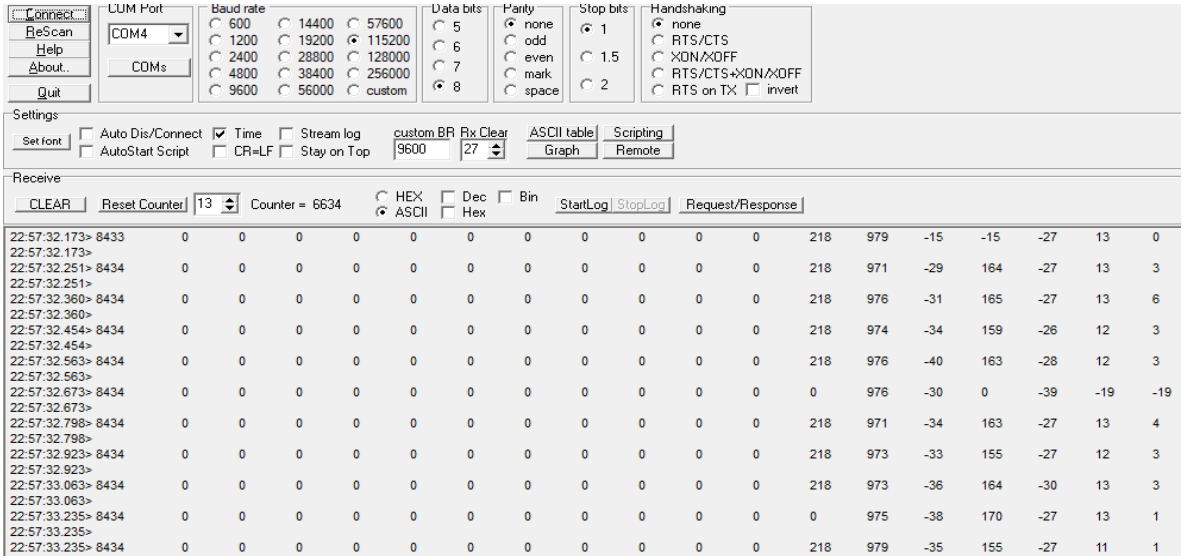


Рис. 4. Результат роботи системи з під'єднаними I2C датчиками

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою роботи є: інтеграція модуля Lilygo LORA32 у багатоканальну радіотехнічну систему на основі FPGA для частотних перетворювачів фізичних величин [4]. Це забезпечить можливість попередньої обробки даних, їх компактне пакування та передачу через бездротові канали зв'язку, такі як Lora, WiFi та Bluetooth. Оскільки об'єм роботи великий, то у даній частині буде розглянуто налаштування середовища розробки, створення драйвера необхідного для взаємодії з багатоканальною радіотехнічною системою на основі FPGA, модуль обробки отриманих даних з FPGA і формування нового пакету даних для подальшої

передачі через LoRa, WiFi, Bluetooth. Це відкриє можливість передачі виміряних даних безпровідним шляхом.

ТЕОРЕТИЧНІ ТА ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

Модуль Lilygo LORA32 [10] побудований на базі мікроконтролера ESP32, який оснащений двоядерним процесором Tensilica LX6 з частотою до 240 МГц. Це забезпечує високу продуктивність для обробки даних та виконання складних алгоритмів. Вбудована флеш-пам'ять і SRAM дозволяють зберігати програми та обробляти великі обсяги даних, а наявність інтерфейсів UART, I2C, SPI, ADC, DAC забезпечує гнучкість у підключенні різних сенсорів рис. 5.

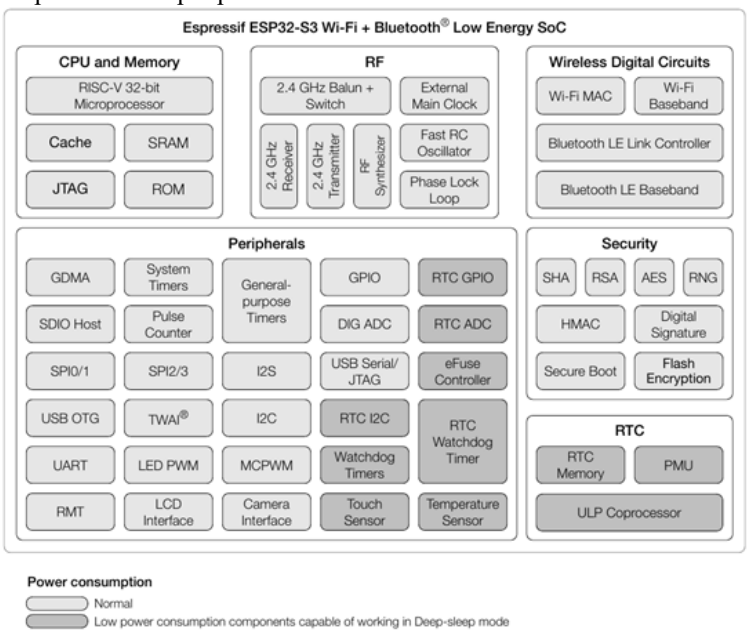


Рис. 5. Блок-схема ESP32-S3

Модуль LoRa використовує чіп SX1276 рис. 6, який підтримує діапазон частот від 137 МГц до 1020 МГц, забезпечуючи довгі дистанції передачі даних з низькою потужністю. Висока чутливість та потужність сигналу дозволяють передавати дані на великі відстані. Lilygo LORA32 підтримує WiFi (IEEE 802.11 b/g/n), а також Bluetooth (BLE та класичний Bluetooth) для низькоенергетичних з'єднань. Інтегрований OLED дисплей дозволяє в реальному часі відображати важливу інформацію та здійснювати відладку системи без необхідності підключення до комп'ютера. Архітектура модуля забезпечує високу продуктивність та гнучкість, що робить його ідеальним для інтеграції у багатоканальні радіотехнічні системи на основі FPGA для частотних перетворювачів фізичних величин.

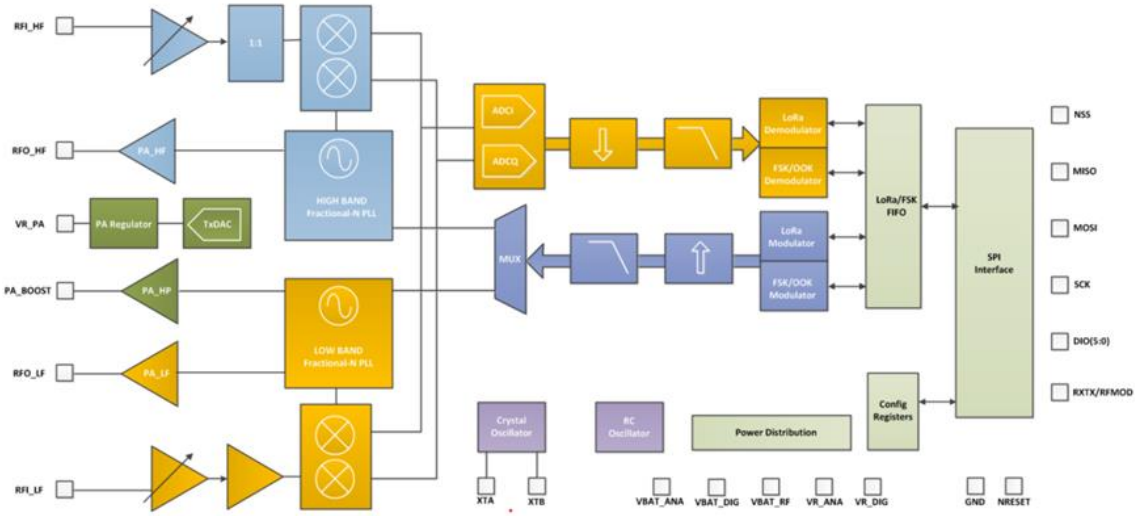


Рис. 6. Блок-схема SX1276

Першим етапом являється налаштування середовище розробки під Lilygo LORA32 і створення проекту. Для пришвидшення етапу розробки було вирішено використовувати Visual Studio Code [11] із плагіном PlatfomIO [12] і фрейм ворком Arduino, рис. 7.

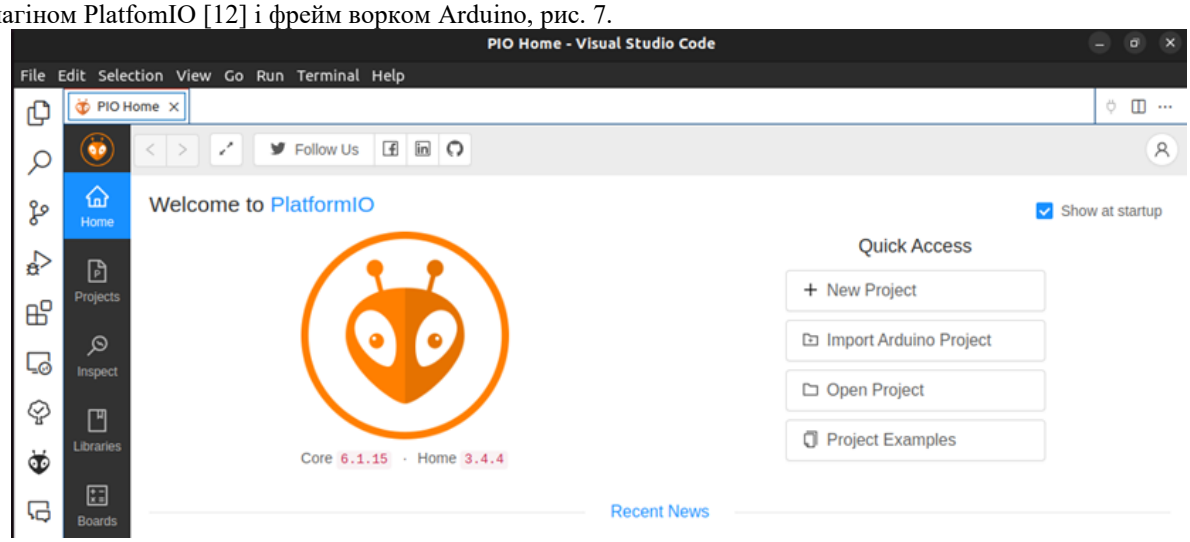


Рис. 7. Visual Studio Code із встановленим PlatfomIO

Створено проект *fpga_meter_rf_unit* із наступними конфігураціями у файлі *platformio.ini*, рис. 8. У налаштуваннях вказано додаткові бібліотеки які будуть використані при роботі із модулем SX1276, SPI і RadioLib. Задано швидкість завантаження прошивки на рівні 921600, і швидкість порту для налагодження 115200. Також змінено конфігурації у файлі *sdkconfig.lilygo-t-display-s3*, для встановлення сумісності із Arduino фрейм ворком, рис. 9.

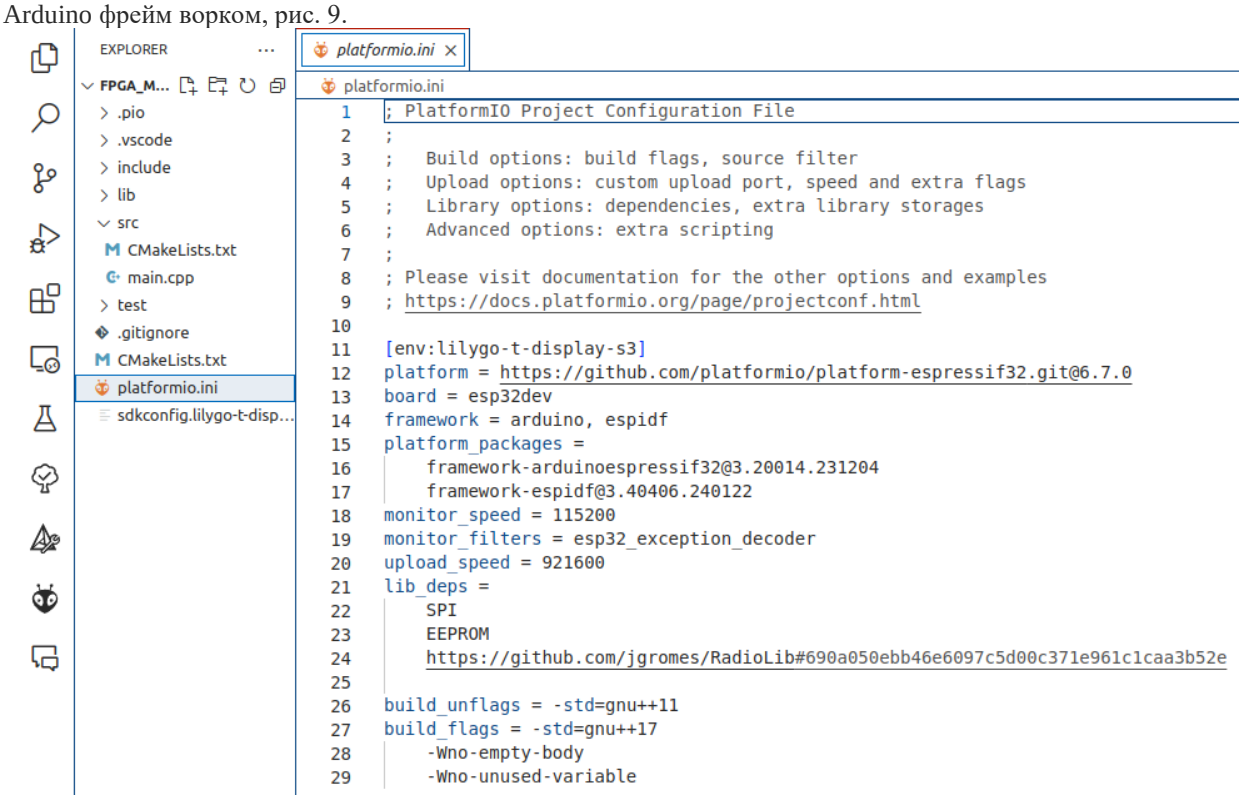


Рис. 8. Налаштування platformio.ini

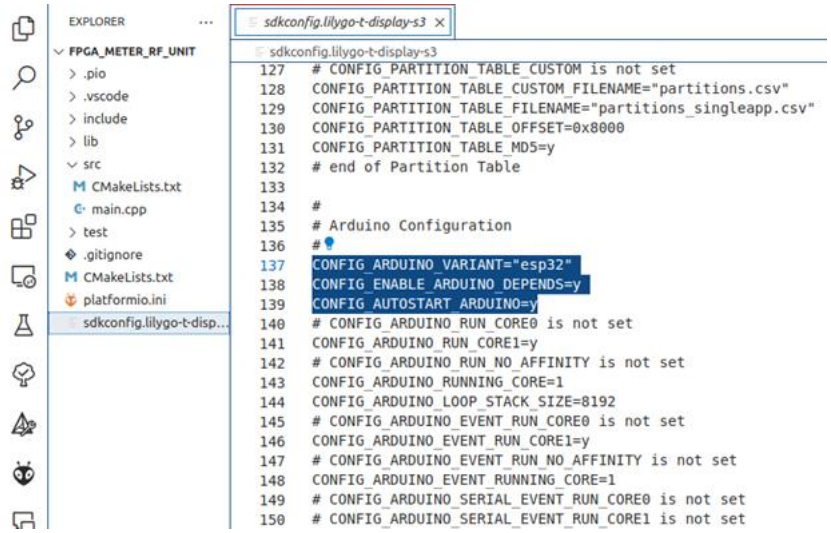


Рис. 9. Змінені налаштування у файлі sdkconfig

Для перевірки працездатності налаштованого середовища було створено простий приклад коду рис. 10, який раз в секунду відправляє на UART повідомлення. Результат роботи відображений на рис. 11.

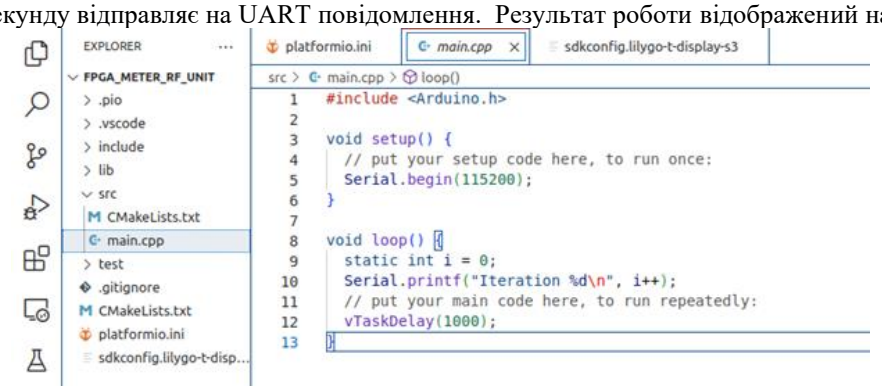


Рис. 10. Налаштування проекту

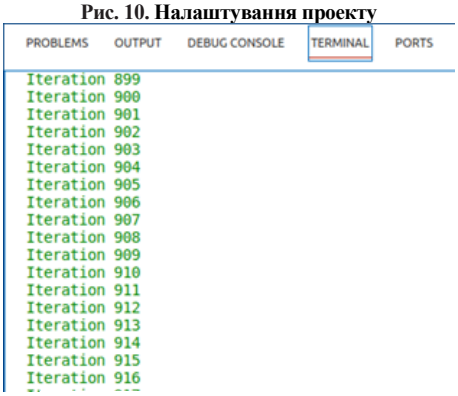


Рис. 11. Результат роботи тестового коду

Другим етапом являється створення драйвера необхідного для взаємодії з багатоканальною радіотехнічною системою на основі FPGA. FPGA використовує UART для передачі виміряних даних, швидкість 115200, кількість біт даних 8, без біта парності, і з одним стоп-бітом. Пакет складається із 12 значень частотомірів, значення температури, значення акселерометра і значення гіроскопа, всі значення розділенні символом /t і записані у вигляді однієї стрічки, між пакетами передається додатковий символ /n. Враховуючи описану структуру було створено *FmUnit* драйвер який зчитує з UART повідомлення, перетворює у структуру і зберігає у чергу. Інтерфейс драйвера разом із вихідною структурою і параметрами зображено на рис. 12.

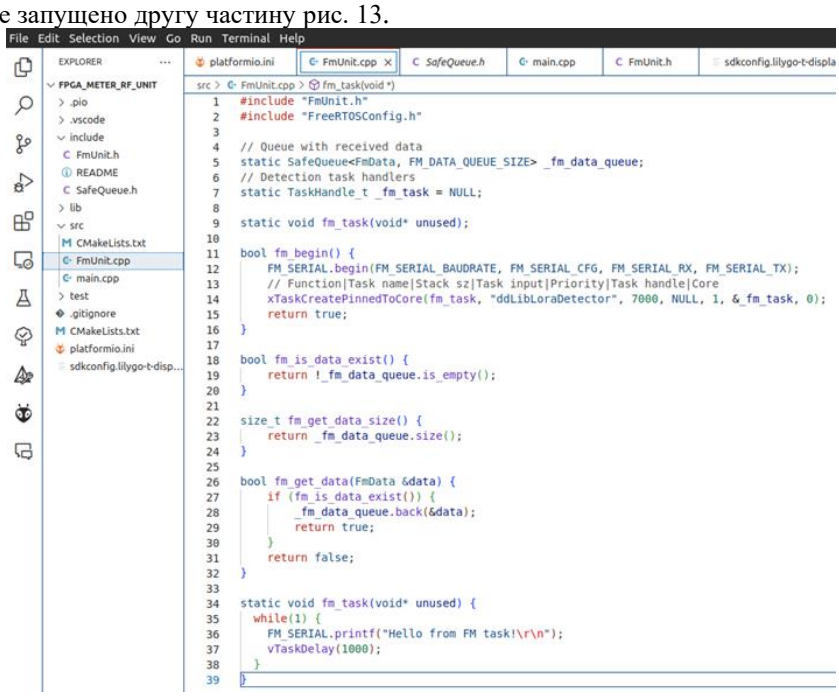
ISSN 2219-9365

include > C FmUnit.h > ...

```
#ifndef _FM_UNIT_H_
#define _FM_UNIT_H_

#include <Arduino.h>
#include "SafeQueue.h"

#define FM_FREQS_CNT 12
#define FM_PARAM_CNT 19

#define FM_FREQ_MIN_ID 0
#define FM_FREQ_MAX_ID 11
#define FM_TEMP_ID 12
#define FM_ACCEL_X_ID 13
#define FM_ACCEL_Y_ID 14
#define FM_ACCEL_Z_ID 15
#define FM_GYRO_X_ID 16
#define FM_GYRO_Y_ID 17
#define FM_GYRO_Z_ID 18

#define FM_DATA_QUEUE_SIZE 50

#define FM_SERIAL Serial1
#define FM_SERIAL_BAUDRATE 115200
#define FM_SERIAL_CFG SERIAL_8N1
#define FM_SERIAL_RX 13
#define FM_SERIAL_TX 15

// To ensure the FmData structure is packed without any additional padding bytes
#pragma pack(push, 1)
struct FmData {
    uint32_t freqs[FM_FREQS_CNT];
    int16_t temp;
    int16_t accel_x;
    int16_t accel_y;
    int16_t accel_z;
    int16_t gyro_x;
    int16_t gyro_y;
    int16_t gyro_z;
};
#pragma pack(pop)

bool fm_begin();
bool fm_is_data_exist();
size_t fm_get_data_size();
bool fm_get_data(FmData &data);

#endif // _FM_UNIT_H_
```

Рис. 12. Інтерфейс драйвера FmUnit

FmUnit складається із двох частин, перша частина ініціалізує UART, черги повідомлень і створює потік у якому буде запущено другу частину рис. 13.




```
#include "FmUnit.h"
#include "FreeRTOSConfig.h"

// Queue with received data
static SafeQueue<FmData, FM_DATA_QUEUE_SIZE> _fm_data_queue;
// Detection task handlers
static TaskHandle_t _fm_task = NULL;

static void fm_task(void* unused);

bool fm_begin() {
    FM_SERIAL.begin(FM_SERIAL_BAUDRATE, FM_SERIAL_CFG, FM_SERIAL_RX, FM_SERIAL_TX);
    // Function|Task name|Stack sz|Task input|Priority|Task handle|Core
    xTaskCreatePinnedToCore(fm_task, "ddLibLoraDetector", 7000, NULL, 1, &_fm_task, 0);
    return true;
}

bool fm_is_data_exist() {
    return !_fm_data_queue.is_empty();
}

size_t fm_get_data_size() {
    return _fm_data_queue.size();
}

bool fm_get_data(FmData &data) {
    if (fm_is_data_exist()) {
        _fm_data_queue.back(&data);
        return true;
    }
    return false;
}

static void fm_task(void* unused) {
  while(1) {
    FM_SERIAL.printf("Hello from FM task!\r\n");
    vTaskDelay(1000);
  }
}
```

Рис. 14. Код ініціалізації драйвера FmUnit

У другій частині відбувається зчитування даних з UART порту рис. 15, і конвертування отриманого пакету у структуру *FmData* рис. 16.

```
39  static void fm_task(void* unused) {
40    while(1) {
41      static std::vector<uint8_t> buffer;
42      if (FM_SERIAL.available() > 0) {
43        uint8_t byte = FM_SERIAL.read();
44        if (byte == '\n') {
45          FmData tmpData;
46          // Parse received data
47          if (fm_parse_data(buffer, tmpData)) {
48              // If queue is full, pop the oldest element
49              if (_fm_data_queue.is_full()) _fm_data_queue.pop();
50              // Push new data to the queue
51              _fm_data_queue.push(&tmpData);
52              Serial.println("New data pushed to the queue");
53          } else {
54              Serial.println("Received damaged data");
55          }
56          // Clear buffer
57          buffer.clear();
58        } else {
59          buffer.push_back(byte);
60        }
61      }
62      vTaskDelay(10);
63    }
64  }
```

Рис. 15. Імплементація потоку для зчитування даних з UART

```
66  static bool fm_parse_data(const std::vector<uint8_t> &buffer, FmData &data) {
67      std::string input(buffer.begin(), buffer.end());
68      std::istringstream stream(input);
69      std::string token;
70      int count = 0;
71
72      while (std::getline(stream, token, '\t')) {
73          // Ensure the token contains only digits or a possible leading '-'
74          if (token.find_first_not_of("-0123456789") != std::string::npos) {
75              return false;
76          }
77
78          int value;
79          std::istringstream value_stream(token);
80          value_stream >> value;
81
82          value *= 10;
83
84          if (count < FM_FREQ_MAX_ID) {
85              data.freqs[count] = value;
86          } else if (count == FM_TEMP_ID) {
87              data.temp = value * 10;
88          } else if (count == FM_ACCEL_X_ID) {
89              data.accel_x = value;
90          } else if (count == FM_ACCEL_Y_ID) {
91              data.accel_y = value;
92          } else if (count == FM_ACCEL_Z_ID) {
93              data.accel_z = value;
94          } else if (count == FM_GYRO_X_ID) {
95              data.gyro_x = value;
96          } else if (count == FM_GYRO_Y_ID) {
97              data.gyro_y = value;
98          } else if (count == FM_GYRO_Z_ID) {
99              data.gyro_z = value;
100         } else {
101             return false; // Too many fields
102         }
103         ++count;
104     }
105     // Ensure we have exactly the expected number of fields
106     return count == FM_PARAM_CNT;
107 }
```

Рис. 16. Конвертування вимірювань FPGA у структуру FmData

На виході драйвера *FmUnit* ми отримуємо компактне представлення даних у вигляді структури *FmData*, дана структура не потребує додаткового форматування, що дозволяє передавати її як корисне навантаження у протоколах LoRa, WiFi і Bluetooth.

Третім етапом являється створення прикладу використання драйвера *FmUnit*, для цього було оновлено код у файлі *main.cpp* рис. 17. Цей код несе виключно інформативну цінність і показує яким чином можуть бути отримані дані. Результат роботи зображений на рис. 18.

src > main.cpp > loop()

```cpp
#include <Arduino.h>
#include "FmUnit.h"

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  Serial.println("RF unit");

  // Initialize FM unit
  if (fm_begin()) {
    Serial.println("FM unit initialized");
  } else {
    Serial.println("FM unit initialization failed");
  }
}

void loop() {
  static int i = 0;
  Serial.printf("Iteration %d\n", i++);

  // Check if new data is available
  if (fm_is_data_exist()) {
    FmData data;
    // Get data from the queue
    if (fm_get_data(data)) {
      Serial.printf("Freqs: %d %d %d %d %d %d %d %d %d %d %d %d\n",
        data.freqs[0], data.freqs[1], data.freqs[2], data.freqs[3], data.freqs[4], data.freqs[5],
        data.freqs[6], data.freqs[7], data.freqs[8], data.freqs[9], data.freqs[10], data.freqs[11]);
      Serial.printf("Temp: %d\n", data.temp);
      Serial.printf("Accel: %d %d %d\n", data.accel_x, data.accel_y, data.accel_z);
      Serial.printf("Gyro: %d %d %d\n", data.gyro_x, data.gyro_y, data.gyro_z);
    }
  }

  // put your main code here, to run repeatedly:
  vTaskDelay(1000);
}
```

Рис. 17. Приклад використання FmUnit

```
Freqs: 84340 0 0 0 0 0 0 0 0 0 0 0
Temp: 21800
Accel: 9760 -15 15
Gyro: -27 13 0
Iteration 58
Freqs: 84340 0 0 0 0 0 0 0 0 0 0 0
Temp: 21800
Accel: 9760 -29 164
Gyro: -27 13 3
Iteration 59
Freqs: 84330 0 0 0 0 0 0 0 0 0 0 0
Temp: 21800
Accel: 9790 -15 15
Gyro: -27 13 0
Iteration 60
Freqs: 84340 0 0 0 0 0 0 0 0 0 0 0
Temp: 21800
Accel: 9710 -29 164
Gyro: -27 13 3
Iteration 61
Freqs: 84340 0 0 0 0 0 0 0 0 0 0 0
Temp: 21800
Accel: 9760 -31 165
Gyro: -27 13 6
Iteration 62
Freqs: 84340 0 0 0 0 0 0 0 0 0 0 0
Temp: 21800
Accel: 9740 -31 165
Gyro: -27 13 6
Iteration 63
Freqs: 84340 0 0 0 0 0 0 0 0 0 0 0
Temp: 21800
Accel: 9760 -29 164
Gyro: -27 13 3
```

Рис. 18. Результат роботи розробленого пристрою

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У цій роботі представлено інтеграцію модуля Lilygo LORA32 у багатоканальну радіотехнічну систему на основі FPGA для частотних перетворювачів фізичних величин. Основним завданням було забезпечення можливості попередньої обробки та бездротової передачі даних через канали зв'язку Lora, WiFi та Bluetooth. Розроблено та налаштовано середовище розробки за допомогою VS Code та PlatformIO, створено драйвер UART для зчитування посилок, алгоритми парсингу отриманих даних, обробки та перевірки помилок, а також методи формування компактних посилок для подальшої передачі. Результати експериментальних досліджень показали високу ефективність та надійність запропонованої системи. Модуль Lilygo LORA32 продемонстрував свою універсальність та продуктивність завдяки підтримці трьох основних протоколів зв'язку, низькому енергоспоживанню та високій чутливості сигналу. Це дозволяє

ефективно використовувати його в різних умовах та застосуваннях, де необхідна стабільна та надійна передача даних на великі відстані. Інтеграція модуля Lilygo LORA32 з FPGA дозволяє значно розширити функціональні можливості вимірювальних систем, підвищуючи їх продуктивність та гнучкість. Це відкриває нові перспективи для розвитку інтелектуальних пристроїв та систем, що забезпечують високоточні вимірювання та бездротову передачу даних.

Література

1. Кофанов В. Л. Лабораторний практикум з дослідження цифрових пристроїв на основі САПР MAX+PLUS II [Текст] : лабораторний практикум / В. Л. Кофанов, О. В. Осадчук, Д. В. Гаврілов. – Вінниця : УНІВЕРСУМ-Вінниця, 2006. – 200 с.
2. Осадчук О.В. Багатоканальний частотомір на програмованій логічній інтегральній схемі для радіовимірювальної системи з частотними сенсорами фізичних величин / Осадчук О.В., Осадчук Я.О., Скощук В.К. //Вісник Хмельницького національного університету, No6, 2021 (303) –С.186-194. DOI 10.31891/2307-5732-2021-303-6-186-194. https://www.doi.org/10.31891/2307-5732-2021-303-6-186-194
3. Осадчук О.В. Використання ядра NIOS II у багатоканальному частотомірі на FPGA для радіотехнічної системи з частотними сенсорами фізичних величин / Осадчук О.В., Осадчук Я.О., Скощук В.К. // Вимірювальна та обчислювальна техніка в технологічних процесах, №1, 2023. – С.137-148. https://doi.org/10.31891/2219-9365-2023-73-1-19
4. Осадчук О.В. Удосконалення багатоканальної радіотехнічної системіна FPGA для частотних перетворювачів фізичних величин підтримкою цифрових сенсорів / Осадчук О.В., Осадчук Я.О., Скощук В.К. // Вимірювальна та обчислювальна техніка в технологічних процесах, №2, 2023. – С.72-82. https://doi.org/10.31891/2219-9365-2023-74-10
5. Nios II Processor Reference Handbook. – San Jose: Altera, 2016. – 260 с.
6. Borgonovo D. Application of the NIOS II processor-FPGA on the digital control of a single-phase PFC rectifier," / D. Borgonovo, M. L. Heldwein and S. A. Mussa // *2008 11th Workshop on Control and Modeling for Power Electronics*, Zurich, Switzerland, 2008, pp. 1-7, https://doi.org/10.1109/COMPEL.2008.4634702
7. Safarpour M. An Embedded Programmable Processor for Compressive Sensing Applications," / M. Safarpour, I. Hautala and O. Silvén // *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)*, Tallinn, Estonia, 2018, pp. 1-5, doi: https://doi.org/10.1109/NORCHIP.2018.8573494
8. Tayara H. A Real-Time Marker-Based Visual Sensor Based on a FPGA and a Soft Core Processor / Tayara H., Ham W., Chong K.T. // *Sensors.* 2016; 16(12):2139. https://doi.org/10.3390/s16122139
9. Magdaleno E. A FPGA Embedded Web Server for Remote Monitoring and Control of Smart Sensors Networks / Magdaleno E., Rodríguez M., Pérez F., Hernández D., García E. // *Sensors.* 2014; 14(1):416-430. https://doi.org/10.3390/s140100416
10. LoRa32 V2.1_1.6 – LILYGO. URL: https://www.lilygo.cc
11. Visual Studio Code. URL: https://code.visualstudio.com/download
12. PlatformIO. URL: https://platformio.org/

References

1. Kofanov, V. L., Osadchuk, O. V., & Havrilov, D. V. (2006). Laboratory Practicum on the Study of Digital Devices Based on CAD MAX+PLUS II [Text]: Laboratory Practicum. Vinnytsia: UNIVERSUM-Vinnytsia. 200 pages.
2. Osadchuk, O. V., Osadchuk, Y. O., & Skoshchuk, V. K. (2021). Multichannel Frequency Meter on a Programmable Logic Device for a Radio Measurement System with Frequency Sensors of Physical Quantities. Bulletin of Khmelnytskyi National University, No. 6, 2021 (303), pp. 186-194. DOI: 10.31891/2307-5732-2021-303-6-186-194. https://www.doi.org/10.31891/2307-5732-2021-303-6-186-194
3. Osadchuk, O. V., Osadchuk, Y. O., & Skoshchuk, V. K. (2023). Using the NIOS II Core in a Multichannel Frequency Meter on FPGA for a Radio Engineering System with Frequency Sensors of Physical Quantities. Measurement and Computing Equipment in Technological Processes, No. 1, 2023, pp. 137-148. https://doi.org/10.31891/2219-9365-2023-73-1-19
4. Osadchuk, O. V., Osadchuk, Y. O., & Skoshchuk, V. K. (2023). Improvement of a Multichannel Radio Engineering System on FPGA for Frequency Converters of Physical Quantities with Support for Digital Sensors. Measurement and Computing Equipment in Technological Processes, No. 2, 2023, pp. 72-82. https://doi.org/10.31891/2219-9365-2023-74-10
5. Nios II Processor Reference Handbook. San Jose: Altera, 2016. 260 pages.
6. Borgonovo, D., Heldwein, M. L., & Mussa, S. A. (2008). Application of the NIOS II Processor-FPGA on the Digital Control of a Single-Phase PFC Rectifier. 2008 11th Workshop on Control and Modeling for Power Electronics, Zurich, Switzerland, pp. 1-7. doi: 10.1109/COMPEL.2008.4634702.
7. Safarpour, M., Hautala, I., & Silvén, O. (2018). An Embedded Programmable Processor for Compressive Sensing Applications. 2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC), Tallinn, Estonia, pp. 1-5. doi: 10.1109/NORCHIP.2018.8573494.
8. Tayara, H., Ham, W., & Chong, K. T. (2016). A Real-Time Marker-Based Visual Sensor Based on an FPGA and a Soft Core Processor. Sensors, 16(12), 2139. https://doi.org/10.3390/s16122139
9. Magdaleno, E., Rodríguez, M., Pérez, F., Hernández, D., & García, E. (2014). A FPGA Embedded Web Server for Remote Monitoring and Control of Smart Sensors Networks. Sensors, 14(1), 416-430. https://doi.org/10.3390/s140100416
10. LoRa32 V2.1_1.6 – LILYGO. URL: https://www.lilygo.cc
11. Visual Studio Code. URL: https://code.visualstudio.com/download
12. PlatformIO. URL: https://platformio.org/