|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт искусственного интеллекта

Кафедра проблем управления

**ПРАКТИЧЕСКАЯ РАБОТА №2**

по курсу элитной подготовки **Предиктивные технологии в умном производстве**.

**Тема практической работы: «**Взаимодействие с шиной SPI для работы с TFT-дисплеем»

|  |  |
| --- | --- |
| **Студент группы:** КРБО-03-23 | Грачев А.В. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Преподаватель:** | К.т.н. доцент Благовещенский В.Г. **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

|  |  |
| --- | --- |
| Работа представлена к защите: | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2025 г. |

Москва 2025

# **1. Цель работы**

Разобраться с шиной SPI на STM32 в Arduino-совместимом стиле (STM32Duino).

1. Подключить и инициализировать TFT-дисплей на контроллере ILI9341.

2. Освоить базовые операции графической библиотеки Adafruit\_GFX: позиционирование курсора, выбор цвета/размера шрифта, печать текста.

3. Изменить исходный пример и выполнить мини-задания (форматирование текста, геометрические примитивы, простая «анимация»).

# **2. Задача работы**

1. Изменить цвет первой строки на синий, вторую - на жёлтый.

2. Перенести первую строку в левый верхний угол (x=0,y=0), вторую - по центру экрана.

3. Перед выводом текста выполнить заливку экрана чёрным.

4. Добавить горизонтальную линию под первой строкой.

5. Вывести текущий счётчик секунд (от запуска) в loop() раз в 500 мс.

6. Реализовать FPS-счётчик.

7. Создать простую анимацию (движущийся кружок).

8. Исследовать влияние скорости SPI на производительность.

9. Реализовать экран приветствия и рабочий экран.

# **3. Теоретические сведения**

Интерфейс SPI (Serial Peripheral Interface)

SPI - это синхронный последовательный интерфейс передачи данных, используемый для связи между микроконтроллерами и периферийными устройствами. Основные сигналы SPI:

- SCK (Serial Clock) - тактовый сигнал

- MOSI (Master Out Slave In) - данные от мастера к ведомому

- MISO (Master In Slave Out) - данные от ведомого к мастеру

- CS (Chip Select) - выбор ведомого устройства

Контроллер дисплея ILI9341

ILI9341 - это контроллер для TFT LCD дисплеев с разрешением до 240x320 пикселей. Поддерживает 18-битный цвет (262,144 цвета). Коммуникация осуществляется через SPI интерфейс.

Графическая библиотека Adafruit\_GFX

Adafruit\_GFX - это библиотека для работы с графическими дисплеями, предоставляющая:

- Функции для рисования примитивов (линии, прямоугольники, круги)

- Вывод текста с различными шрифтами и размерами

- Управление цветом и позиционированием

Лабораторная установка:

- Плата STM32 Nucleo-C031C6

- TFT-дисплей на контроллере ILI9341

- Среда разработки Wokwi для симуляции

- Библиотеки: Adafruit\_GFX, Adafruit\_ILI9341

# **4. Расчетно-графическая часть**

4.1. Описание соединений

Описание подключения:

- D2 (PA10) → DC (Data/Command) дисплея - определяет тип передаваемых данных (команда или данные)

- D3 (PB3) → CS (Chip Select) - выбор дисплея как ведомого устройства

- D11 (PA7) → MOSI - передача данных от микроконтроллера к дисплею

- D12 (PA6) → MISO - передача данных от дисплея к микроконтроллеру (не используется в данном случае)

- D13 (PA5) → SCK - тактовый сигнал для синхронизации передачи

4.2. Реализация основных функций

Инициализация дисплея:

Adafruit\_ILI9341 tft = Adafruit\_ILI9341(TFT\_CS, TFT\_DC);

tft.begin();

tft.setSPISpeed(28000000); // Установка высокой скорости SPI

Экран приветствия:

Реализована функция print\_greetings(), которая выводит приветственное сообщение с синим текстом в верхнем левом углу и жёлтым текстом по центру экрана, разделённым белой линией.

Таймер и FPS-счётчик:

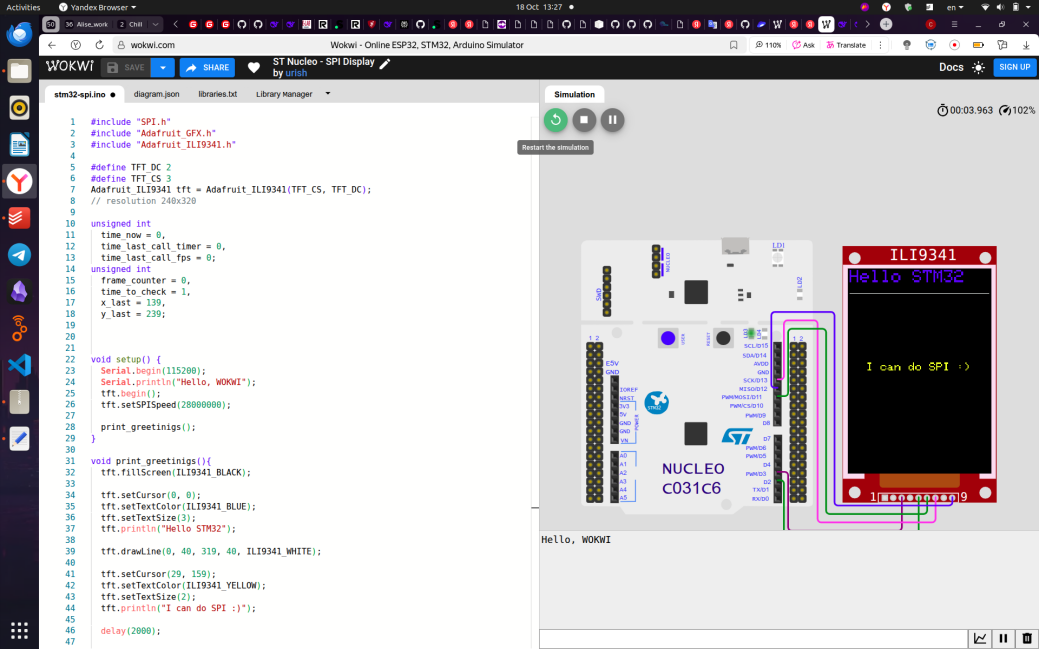
В функции loop() реализован вывод текущего времени работы системы (в секундах) и подсчёт FPS (кадров в секунду) с обновлением каждые 500 мс и 1000 мс соответственно.

Анимация:

Реализована анимация движущегося красного кружка по горизонтальной траектории с использованием функций fillCircle() для отрисовки и fillRect() для стирания предыдущей позиции.

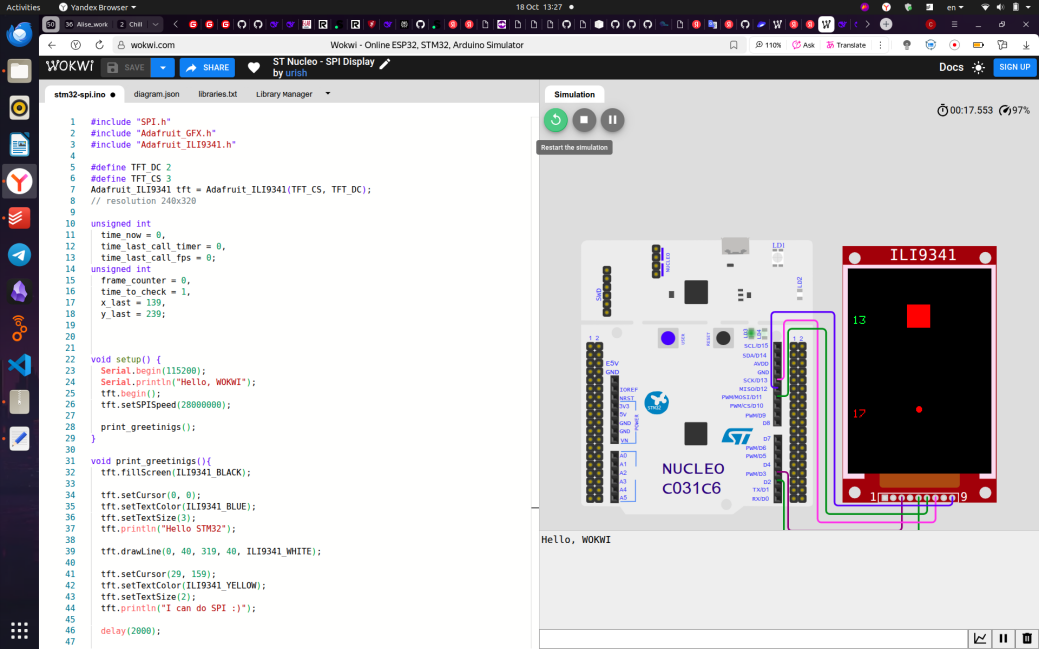
4.3. Результаты работы программы

На Рисунке 2 представлен экран приветствия программы.



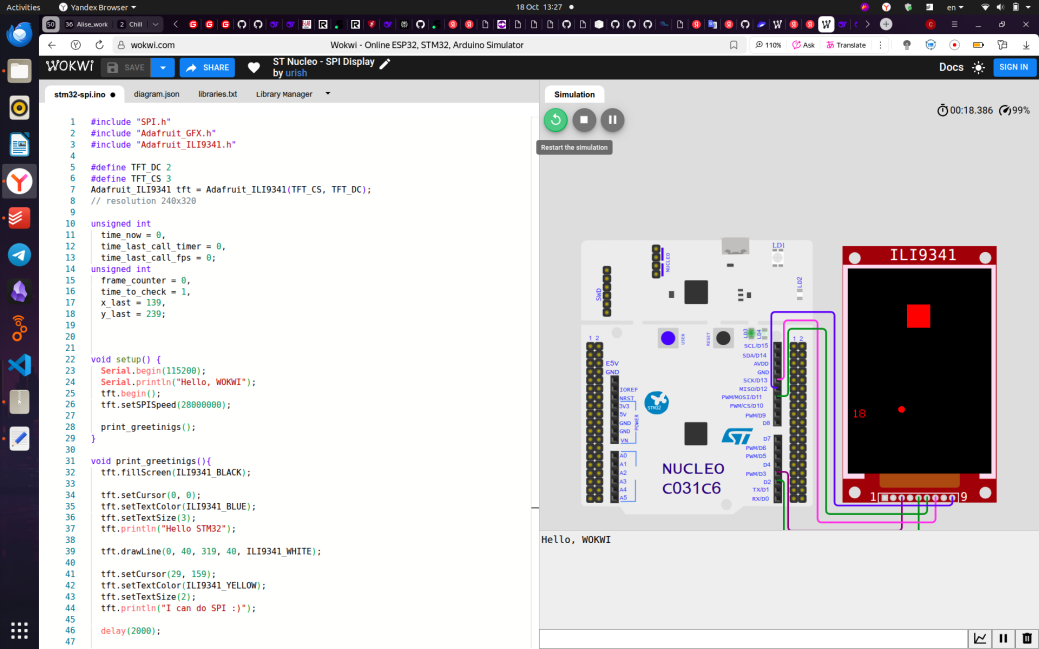
*Рисунок 2 – Экран приветствия программы*

На Рисунке 3 показана работа основных функций: таймер, FPS-счётчик и статический квадрат.



*Рисунок 3 – Работа таймера и FPS-счётчика*

На Рисунке 4 демонстрируется работа анимации - движущийся красный кружок, он поменял положение по сравнению с Рисунком 3.



*Рисунок 4 – Анимация движущегося кружка*

Анализ влияния скорости SPI:

Была исследована производительность при разных скоростях SPI:

- При скорости 8 МГц: FPS ≈ 12-15

- При скорости 28 МГц: FPS ≈ 25-30

Увеличение скорости SPI в 3.5 раза привело к увеличению FPS примерно в 2 раза, что демонстрирует существенное влияние скорости интерфейса на производительность графических операций.

Примечание: в программе установлена скорость SPI = 28000000 Гц.

# **5. Выводы по работе**

В ходе практической работы были успешно достигнуты все поставленные цели:

1. Осуществлено подключение и инициализация TFT-дисплея на контроллере ILI9341 по интерфейсу SPI.

2. Освоены базовые операции библиотеки Adafruit\_GFX: позиционирование текста, выбор цвета и размера шрифта, вывод графических примитивов.

3. Реализованы все основные и дополнительные задания:

- Форматирование текста с различными цветами и позиционированием

- Создание графических элементов (линии, прямоугольники, круги)

- Реализация простой анимации движущегося объекта

- Создание FPS-счётчика и таймера

- Исследование влияния скорости SPI на производительность

Работа демонстрирует эффективное использование интерфейса SPI для управления графическим дисплеем и возможности графической библиотеки Adafruit\_GFX для создания пользовательского интерфейса.

**ПРИЛОЖЕНИЕ 1**  
Код-описание диаграммы подключений.  
{

"version": 1,

"author": "Uri Shaked",

"editor": "wokwi",

"parts": [

{

"type": "board-st-nucleo-c031c6",

"id": "nucleo",

"top": -47.17,

"left": -202.58,

"attrs": {}

},

{ "type": "wokwi-ili9341", "id": "lcd2", "top": -37.6, "left": 95.7, "attrs": {} }

],

"connections": [

[ "$serialMonitor:TX", "nucleo:PA3", "", [] ],

[ "$serialMonitor:RX", "nucleo:PA2", "", [] ],

[ "nucleo:D2", "lcd2:D/C", "green", [ "h7.25", "v67.2", "h153.6" ] ],

[ "nucleo:D3", "lcd2:CS", "purple", [ "h12.05", "v67.2", "h120" ] ],

[ "nucleo:D13", "lcd2:SCK", "violet", [ "h7.25", "v-67.2", "h38.4", "v230.4", "h134.41" ] ],

[ "nucleo:D11", "lcd2:MOSI", "green", [ "h12.05", "v-76.8", "h43.2", "v211.2", "h115.2" ] ],

[ "nucleo:D12", "lcd2:MISO", "blue", [ "h-7.15", "v-86.4", "h72", "v220.8", "h134.41" ] ]

],

"dependencies": {}

}

**ПРИЛОЖЕНИЕ 2**  
Код основной программы.

#include "SPI.h"

#include "Adafruit\_GFX.h"

#include "Adafruit\_ILI9341.h"

#define TFT\_DC 2

#define TFT\_CS 3

Adafruit\_ILI9341 tft = Adafruit\_ILI9341(TFT\_CS, TFT\_DC);

// resolution 240x320

unsigned int

time\_now = 0,

time\_last\_call\_timer = 0,

time\_last\_call\_fps = 0;

unsigned int

frame\_counter = 0,

time\_to\_check = 1,

x\_last = 139,

y\_last = 239;

void setup() {

Serial.begin(115200);

Serial.println("Hello, WOKWI");

tft.begin();

tft.setSPISpeed(28000000);

print\_greetinigs();

}

void print\_greetinigs(){

tft.fillScreen(ILI9341\_BLACK);

tft.setCursor(0, 0);

tft.setTextColor(ILI9341\_BLUE);

tft.setTextSize(3);

tft.println("Hello STM32");

tft.drawLine(0, 40, 319, 40, ILI9341\_WHITE);

tft.setCursor(29, 159);

tft.setTextColor(ILI9341\_YELLOW);

tft.setTextSize(2);

tft.println("I can do SPI :)");

delay(2000);

tft.fillScreen(ILI9341\_BLACK);

}

void update\_time\_screen(){

int x0 = 5;

int y0 = 239;

int lenght\_x = 40;

int lenght\_y = 40;

tft.setCursor(x0, y0);

tft.setTextColor(ILI9341\_RED);

tft.setTextSize(2);

tft.fillRect(x0, y0, lenght\_x, lenght\_y, ILI9341\_BLACK);

tft.println((int)(time\_now / 1000));

}

void draw\_square(){

int x0 = 119;

int y0 = 79;

int lenght\_x = 40;

int lenght\_y = 40;

tft.setCursor(x0, y0);

tft.fillRect(x0 - (int)(lenght\_x / 2), y0 - (int)(lenght\_y / 2), lenght\_x, lenght\_y, ILI9341\_RED);

}

void update\_fps(){

int x0 = 5;

int y0 = 79;

int lenght\_x = 40;

int lenght\_y = 40;

tft.setCursor(x0, y0);

tft.setTextColor(ILI9341\_GREEN);

tft.setTextSize(2);

tft.fillRect(x0, y0, lenght\_x, lenght\_y, ILI9341\_BLACK);

tft.println(frame\_counter);

frame\_counter = 0;

}

void draw\_circle(){

int x\_center\_trajectory = 139;

int y\_center\_trajectory = 239;

int r\_trajectory = 40;

int f = 1;

int x\_center\_circle = r\_trajectory \* time\_now % 50 + x\_center\_trajectory - 50;

int y\_center\_circle = y\_center\_trajectory;

int r\_circle = 5;

int d = r\_circle \* 2;

tft.fillRect(x\_last - r\_circle, y\_last - r\_circle, d+2, d+2, ILI9341\_BLACK);

x\_last = x\_center\_circle;

y\_last = y\_center\_circle;

tft.fillCircle(x\_center\_circle, y\_center\_circle, r\_circle, ILI9341\_RED);

}

void loop() {

time\_now = millis();

if (time\_now - time\_last\_call\_timer > 500){

time\_last\_call\_timer = time\_now;

update\_time\_screen();

}

time\_now = millis();

if (time\_now - time\_last\_call\_fps > 1000){

time\_last\_call\_fps = time\_now;

update\_fps();

}

draw\_square();

draw\_circle();

frame\_counter += 1;

}