Санкт-Петербургский политехнический университет Петра Великого
Институт металлургии, машиностроения и транспорта
Высшая школа автоматизации и робототехники

Курсовая работа

Дисциплина: Программирование на языках высокого уровня
Тема: Создание трехколесного робота

Выполнил студент гр. 3331506/10401 Папасимеониди А.С.

Преподаватель Ананьевский М.С.

«___»_____ 2024 г.

Санкт-Петербург
2024

Содержание

1. Формулировка задачи, которую решает студент

Задача заключается в построении трехколесного робота. Задача студента на начальном этапе создать системы управления микроконтроллера с экраном SSD1306, адресной светодиодной лентой WS2812B, а также двух двигателей.

Ссылка на гит: https://github.com/simeonidi03/hal

2. Среда разработки и микроконтроллер

В связи с изменившимися логистическими цепочками, микроконтроллеры серии STM32 стали недоступны для студенческих проектов в связи с высокой стоимостью (1600 рублей за камень). Для создания проекта был использован аналог STM32 от компании Artery. Студент работает с платой для стартовой отладки (схожа с Arduino), называется плата Artery-START-F413.

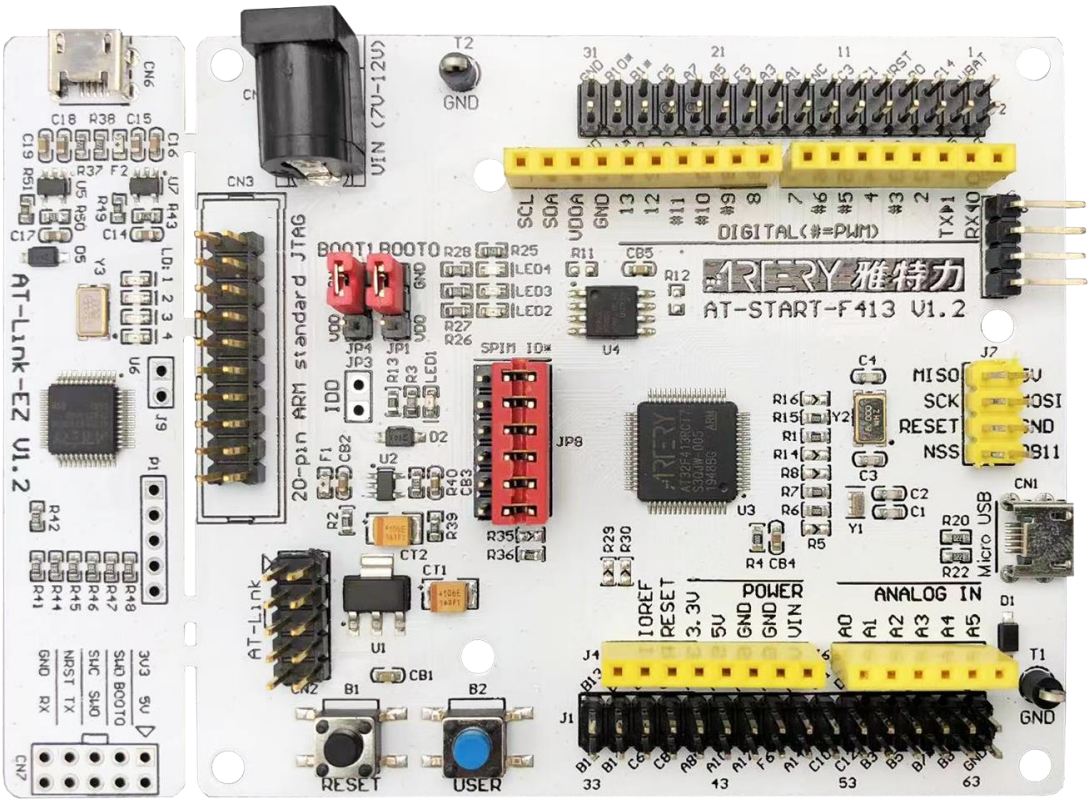

Рисунок 2.1 – Фотография платы

В данную плату встроен программатор (левая часть платы). Работу произвожу в проприетарной IDE под название AT32IDE. Также для программирования данной платы возможно использование Keil 5 или 4 версии. Доступен кодогенератор (аналог CubeMX от STM). IDE базируется на Eclipse.

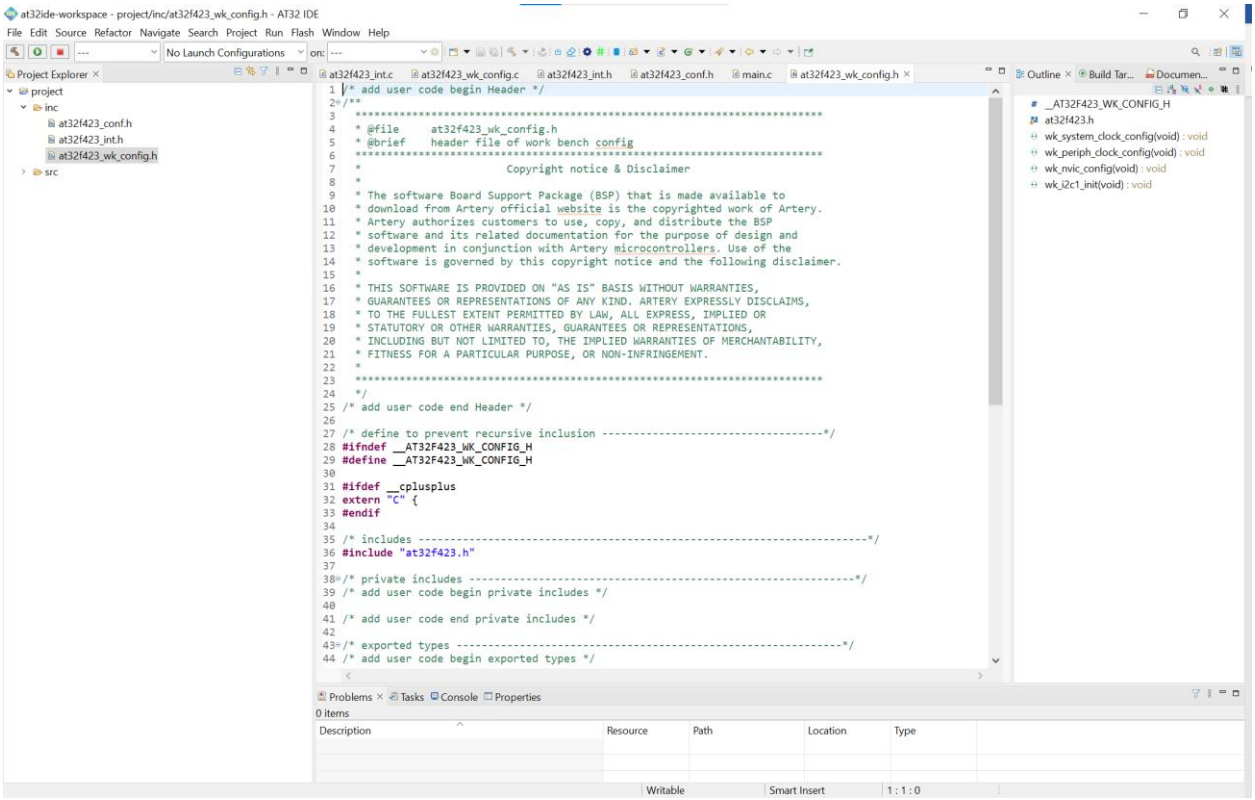

Рисунок 2.2 – AT32IDE

На этом краткое описание среды работы заканчивается.

3. Словесное описание алгоритма работы дисплея SSD1306

Дисплей с микроконтроллером соединяется по шине I2C. В идеологии этой шины есть главное и управляемое устройство. Главным будет являться микроконтроллер, управляемым будет экран. Шина I2C имеет две линии передачи данных.

Линия SCL – линия для тактирования (т.е. управления передачей данных и согласования всех устройств между собой).

Линия SDA – передача данных.

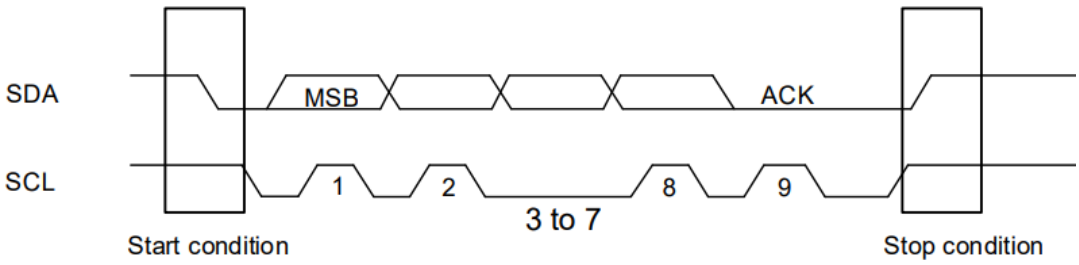

Рисунок 3.1 – Пример передачи данных по I2C

Начальное условие: Когда SCL установлен на высоком уровне, SDA переключается с высокого на низкий

Условие остановки: Когда SCL установлен на высоком уровне, SDA переключается с низкого на высокий.

В данном проекте я не использую прямой доступ к памяти (DMA), так как установка картинки на экран будет производиться в начале, когда процессор не будет загружен другими задачами.

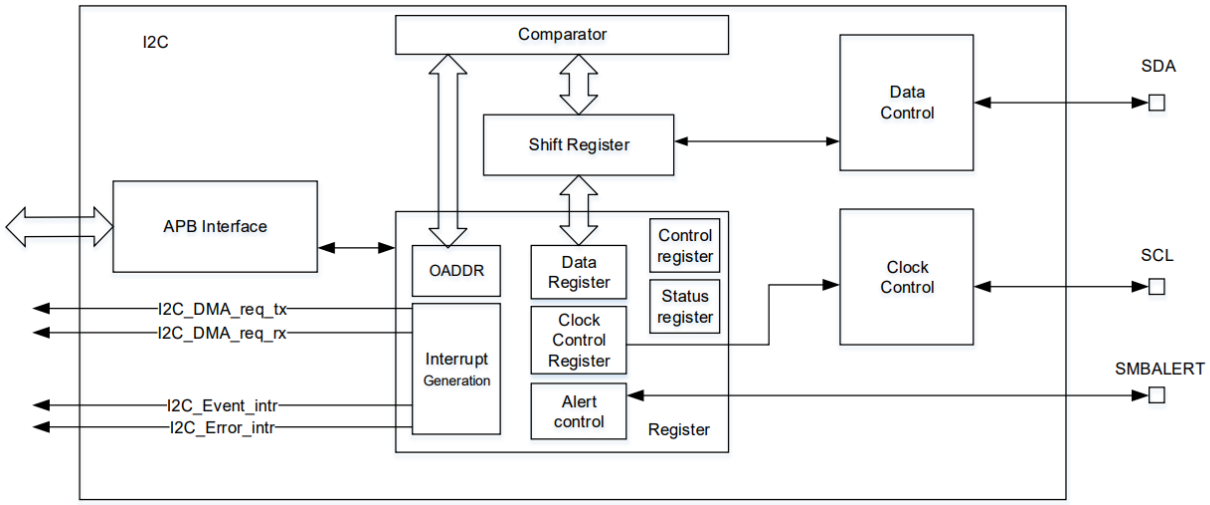

Рисунок 3.2 – Схема работы шины I2C

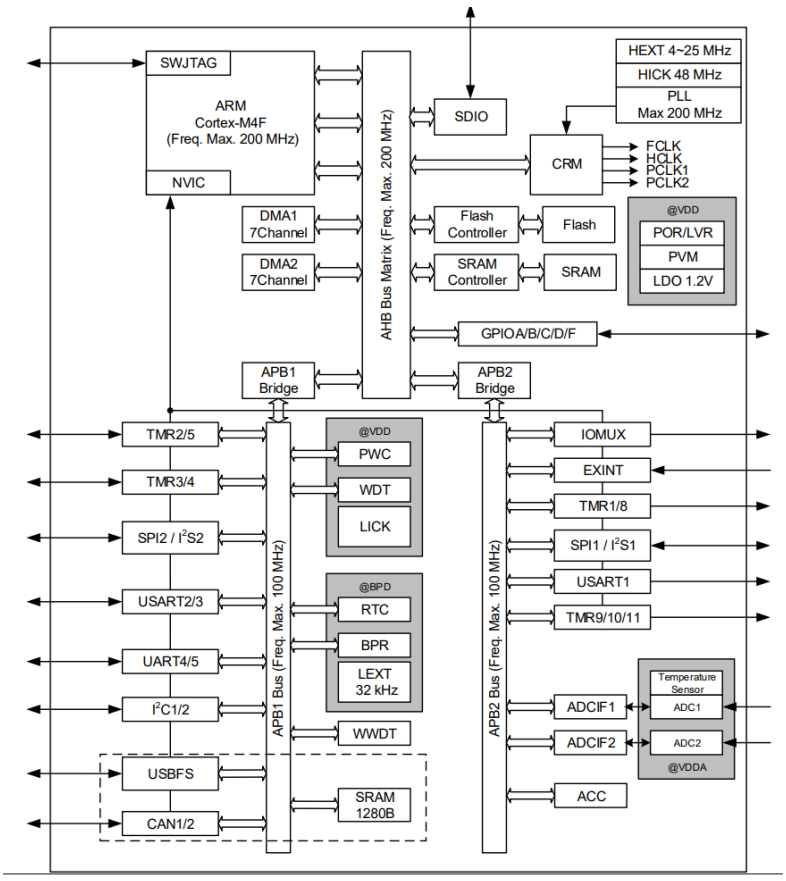

Рисунок 3.3 – Архитектура микроконтроллера

Шина I2C берёт тактирование с шины APB1 через шину AHB. Максимальная частота данной серии микроконтроллеров равна 200 МГц.

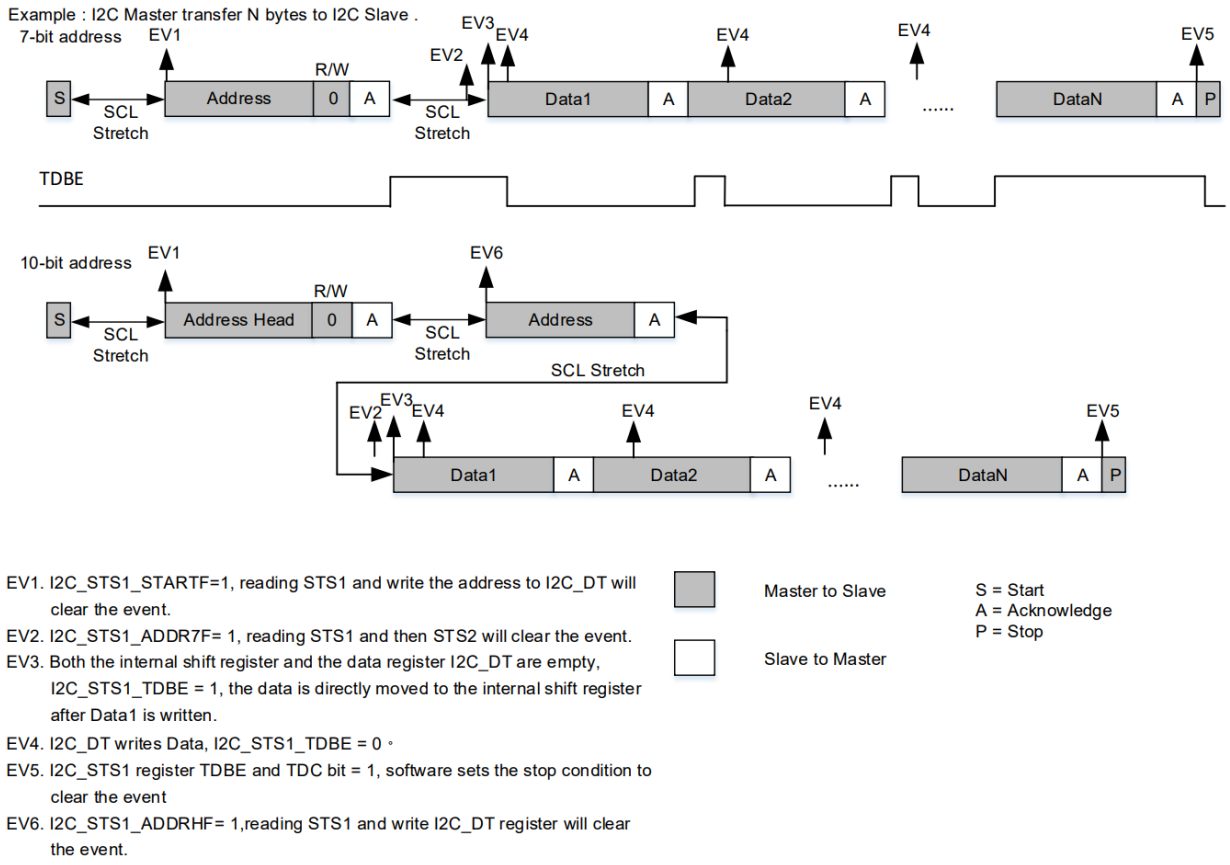

Рисунок 3.4 – Схема работы шины I2C

Сначала в шину передаётся адрес устройства (slave). В нашем случае это 0x3C (16-ричная система).

Также пару слов скажу о библиотеках в экосистеме Artery. Они используют union – объединения. Это участок памяти, который используется несколькими разными переменными, которые могут быть даже различного типа. Часто используются макросы (стандартно для библиотек для встроенных систем). Используются шаблоны структур для упрощения кода (без использования ключевого слова struct).

Ход работы:

- 1. Для использования пинов нужно настроить пин с помощью GPIO. Изначально пытался реализовать с помощью переписывания библиотеки, написанной под STM32, затем изучил аналог библиотеки HAL в Artery, с помощью неё настроил порты 6 и 7 GPIOB на SDA и SCL соответственно. Порты настроены в режиме GPIO_MODE_MUX, что обозначает порты как мультиплексоры. Этот режим позволяет использовать один и

тот же порт GPIO для различных целей, переключаясь между ними с помощью ПО.

- Мы используем I2C1 в основном режиме. Дисплей не будет передавать какие-либо данные на микроконтроллер. I2C будет работать в быстром режиме, 400 кГц.
- Параметр DutyCycle, определяющий коэффициент заполнения, те времени в течении которого сигнал на шине данных будет на высоком уровне, ставим в режим 16/9.
- Также отмечу, что в библиотеке данного микроконтроллера используется идеология структурного выбора, в отличии от STM, где зачастую используют побитовые операции.
- Для начала передачи по I2C мы должны установить бит START (см. рис. 3.4 в единицу). Это запустит передачу данных (при успешном подтверждении адреса Slave устройством).

Для программирования микроконтроллером используют различные идеологии. Любо программирование с помощью встроенных функций (часто используется начинающими), либо программирование на уровне битов.

Рассмотрим регистр ctrl1_bit. Это структура с union внутри. Эта структура позволяет управлять передачей данных по I2C. Она позволяет обращаться к отдельным флагам управления для удобной постановки и снятия флагов настройки I2C. Каждый бит в этой структуре соответствует определенной настройке или флагу управления в I2C, позволяя программисту легко манипулировать отдельными настройками шины через их битовое представление в регистре ctrl1.

Для оценки статуса передачи используют статусный регистр.

4. Реализация работы экрана SSD1306

Листинг программы с реализацией работы дисплея

main.c

```
#include "at32f413_board.h"
#include "at32f413_clock.h"
#include "at32f413_wk_config.h"
#include "ssd1306.h"
```

```
#include <stdio.h>
#include "i2c_at_lib.h"
#include "images.h"
/** @addtogroup AT32F413_periph_template
  * @{
  */

/** @addtogroup 413_LED_toggle LED_toggle
  * @{
  */

#define DELAY                            100
#define FAST                             1
#define SLOW                             4

uint8_t g_speed = FAST;

void button_exint_init(void);
void button_isr(void);

/**
  * @brief  configure button exint
  * @param  none
  * @retval none
  */
void button_exint_init(void)
{
  exint_init_type exint_init_struct;

  crm_periph_clock_enable(CRM_IOMUX_PERIPH_CLOCK, TRUE);
  gpio_exint_line_config(GPIO_PORT_SOURCE_GPIOA, GPIO_PINS_SOURCE0);

  exint_default_para_init(&exint_init_struct);
  exint_init_struct.line_enable = TRUE;
  exint_init_struct.line_mode = EXINT_LINE_INTERRUPUT;
  exint_init_struct.line_select = EXINT_LINE_0;
  exint_init_struct.line_polarity = EXINT_TRIGGER_RISING_EDGE;
  exint_init(&exint_init_struct);

  nvic_priority_group_config(NVIC_PRIORITY_GROUP_4);
  nvic_irq_enable(EXINT0_IRQn, 0, 0);
}

/**
  * @brief  button handler function
  * @param  none
  * @retval none
  */
void button_isr(void)
{
```

```
  /* delay 5ms */
  delay_ms(5);

  /* clear interrupt pending bit */
  exint_flag_clear(EXINT_LINE_0);

  /* check input pin state */
  if(SET == gpio_input_data_bit_read(USER_BUTTON_PORT, USER_BUTTON_PIN))
  {
    if(g_speed == SLOW)
      g_speed = FAST;
    else
      g_speed = SLOW;
  }
}

/**
  * @brief  exint0 interrupt handler
  * @param  none
  * @retval none
  */
void EXINT0_IRQHandler(void)
{
  button_isr();
}

/**
  * @brief  main function.
  * @param  none
  * @retval none
  */
int main(void)
{
  /* add user code begin 1 */

  /* add user code end 1 */

  /* system clock config. */
  wk_system_clock_config();

  /* config periph clock. */
  wk_periph_clock_config();

  /* nvic config. */
  wk_nvic_config();

  /* init i2c1 function. */
  wk_i2c1_init();

  /* init i2c1 function. */
```

```
  uint8_t ssd1306_int_ok = SSD1306_Init();

//  system_clock_config();

  at32_board_init();

  button_exint_init();

  if(ssd1306_int_ok){
         at32_led_toggle(LED4);
  } else{
         at32_led_toggle(LED2);
  }


  SSD1306_PutsE("GEOSCAN,", &Font_16x26, SSD1306_COLOR_WHITE, 0, 0); //пишем
надпись в выставленной позиции шрифтом "Font_7x10" белым цветом.
  SSD1306_UpdateScreen();

  delay_ms(g_speed * DELAY * 10);
  SSD1306_PutsE("RUSSIA,", &Font_7x10, SSD1306_COLOR_WHITE, 10, 31);
  SSD1306_PutsE("St. Peterburg", &Font_7x10, SSD1306_COLOR_WHITE, 10, 41);
  SSD1306_UpdateScreen();

  delay_ms(g_speed * DELAY * 10);
  SSD1306_Fill(SSD1306_COLOR_BLACK);
  SSD1306_Geoscan();
  SSD1306_UpdateScreen();

  delay_ms(g_speed * DELAY * 10);
  SSD1306_Fill(SSD1306_COLOR_BLACK);
  SSD1306_UpdateScreen();


  ssd1306_DrawBitmap(0, 0, image1_bits, 128, 64, SSD1306_COLOR_WHITE);
  SSD1306_UpdateScreen();
  delay_ms(g_speed * DELAY * 10);


  ssd1306_DrawBitmap(0, 0, image1_bits, 128, 64, SSD1306_COLOR_WHITE);
  SSD1306_UpdateScreen();
  delay_ms(g_speed * DELAY * 10);

  SSD1306_Fill(SSD1306_COLOR_BLACK);
  ssd1306_DrawBitmap(0, 0, Bus, 128, 64, SSD1306_COLOR_WHITE);
  SSD1306_UpdateScreen();
  delay_ms(g_speed * DELAY * 10);
  SSD1306_UpdateScreen();
```

```
  //SSD1306_DrawCircle(63, 45, 15, SSD1306_COLOR_WHITE); //рисуем белую
окружность в позиции 10;33 и радиусом 7 пикселей
  //SSD1306_UpdateScreen();
  //SSD1306_Fill(SSD1306_COLOR_WHITE);


  while(1)
  {
    at32_led_toggle(LED2);
    delay_ms(g_speed * DELAY);
    at32_led_toggle(LED3);
    delay_ms(g_speed * DELAY);
    at32_led_toggle(LED4);
    delay_ms(g_speed * DELAY);
  }
}
/*
uint8_t SSD1306_Init(); //Инициализация

SSD1306_UpdateScreen(); //Посылаем данные из буффера в памяти дисплею

SSD1306_ToggleInvert(); //инвертирует цвета изображения в оперативной памяти

SSD1306_Fill(SSD1306_COLOR_t Color); //заполняем дисплей желаемым цветом

SSD1306_DrawPixel(uint16_t x, uint16_t y, SSD1306_COLOR_t color);
//нарисовать один пиксел

SSD1306_GotoXY(uint16_t x, uint16_t y); //установить позицию текстового
курсора

SSD1306_Putc(char ch, FontDef_t* Font, SSD1306_COLOR_t color); //вывести
символ ch в позиции курсора

SSD1306_Puts(char* str, FontDef_t* Font, SSD1306_COLOR_t color); //вывести
строку str в позиции курсора

SSD1306_DrawLine(uint16_t x0, uint16_t y0, uint16_t x1, uint16_t y1,
SSD1306_COLOR_t c); //нарисовать линию

SSD1306_DrawRectangle(uint16_t x, uint16_t y, uint16_t w, uint16_t h,
SSD1306_COLOR_t c); //наррисовать прямоугольник

SSD1306_DrawFilledRectangle(uint16_t x, uint16_t y, uint16_t w, uint16_t h,
SSD1306_COLOR_t c); //заполненный прямоугольник

SSD1306_DrawTriangle(uint16_t x1, uint16_t y1, uint16_t x2, uint16_t y2,
uint16_t x3, uint16_t y3, SSD1306_COLOR_t color); //треугольник
```

```
SSD1306_DrawCircle(int16_t x0, int16_t y0, int16_t r, SSD1306_COLOR_t c);
//круг радиуса r

SSD1306_DrawFilledCircle(int16_t x0, int16_t y0, int16_t r, SSD1306_COLOR_t
c); //заполненный круг
*/
```

5. Анализ работы

6. Применение алгоритма

/*какие-то выводы*/

7. Заключение