Федеральное государственное бюджетное образовательное учреждение
высшего образования «Воронежский государственный университет»

Отчет по лабораторной работе № 5
По курсу «Введение в интернет вещей»

«шина связи I2C»

Выполнила:
Меркутова Кристина Денисовна
3 группа

Воронеж 2025

Цель работы:

Освоить принципы функционирования и программной реализации однопроводной шины данных I2C на микроконтроллере ESP32, включая инициализацию шины, обнаружение устройств, обмен данными с датчиками (например, BMP280).

Ход выполнения работы:

1. Ответьте на вопросы в теоретической части максимально подробно;

2. Соберите схему практической части и напишите программу для микроконтроллера. Убедитесь, что код выполняется верно, используйте симулятор wokwi или лабораторный стенд. Скриншоты или фото устройства и листинг программы микроконтроллера обязательны.

1. Теоретическая часть. Ответьте на вопросы:

1. Какие два обязательных сигнальных провода использует шина I²C и какова их функция?
   Ответ: SDA - линия используется для передачи фактических данных между устройствами на шине. По ней биты данных передаются последовательно, один за другим. SCL - линия используется для синхронизации всех устройств на шине. Ведущее устройство (Master) генерирует тактовый сигнал, который определяет скорость передачи данных. Биты данных на линии SDA считаются валидными только в определенные моменты времени (например, во время низкого или высокого уровня SCL).
2. Как ESP32 поддерживает I²C: аппаратно или программно (bit-banging)? Сколько независимых аппаратных I²C-интерфейсов доступно в ESP32, и можно ли переназначать их на произвольные GPIO;
   Ответ: ESP32 поддерживает I²C как аппаратно, так и программно (bit-banging). В ESP32 доступно два независимых аппаратных I²C-контроллера (порта).
   Они обычно обозначаются как:
   - I2C0
   - I2C1

Эти контроллеры могут работать совершенно независимо друг от друга, каждый со своей скоростью (до 1 МГц в режиме fast-mode plus) и набором подключенных устройств. Можно переназначить пины для аппаратных интерфейсов I²C практически на любые GPIO. Это реализуется с помощью внутренней матрицы переназначения (GPIO matrix), которая позволяет маршрутизировать внутренние сигналы на физические выводы.

3. Как определяется адрес ведомого устройства в I²C? Приведите пример, как узнать 7-битный адрес датчика BME280 и как он передаётся на шине (с учётом бита R/W;

   Ответ: адрес BME280 не является случайным, он указан в документации (datasheet) производителя.

   Согласно даташиту:

   Адрес BME280 зависит от состояния вывода SDO.

   - Если вывод SDO подключен к земле (GND), адрес равен 0x76 (в шестнадцатеричном формате) или 118 в десятичном.
   - Если вывод SDO подключен к питанию (VCC), адрес равен 0x77 (в шестнадцатеричном формате) или 119 в десятичном.

   Или еще практический способ создав скетч в Arduino IDE. На шину он передается первым байтом, который состоит из 7 бит адреса + 1 бит чтения/записи (R/W).

4. Как в ESP-IDF инициализируется I²C-драйвер (функция i2c_param_config())? Какие параметры необходимо задать: частота, GPIO, режим, фильтрация шума? Как частота влияет на максимальную длину шины.

   Ответ: процесс инициализации состоит из двух основных шагов: настройка параметров и установка драйвера.

   i2c_config_t conf = {

   .mode = I2C_MODE_MASTER, *// Режим ведущего*

   .sda_io_num = I2C_MASTER_SDA_IO, *// Пин SDA*

   .scl_io_num = I2C_MASTER_SCL_IO, *// Пин SCL*

   .sda_pullup_en = GPIO_PULLUP_ENABLE, *// Подтяжка SDA к VCC*

   .scl_pullup_en = GPIO_PULLUP_ENABLE, *// Подтяжка SCL к VCC*

   .master.clk_speed = I2C_MASTER_FREQ_HZ, *// Тактовая частота*

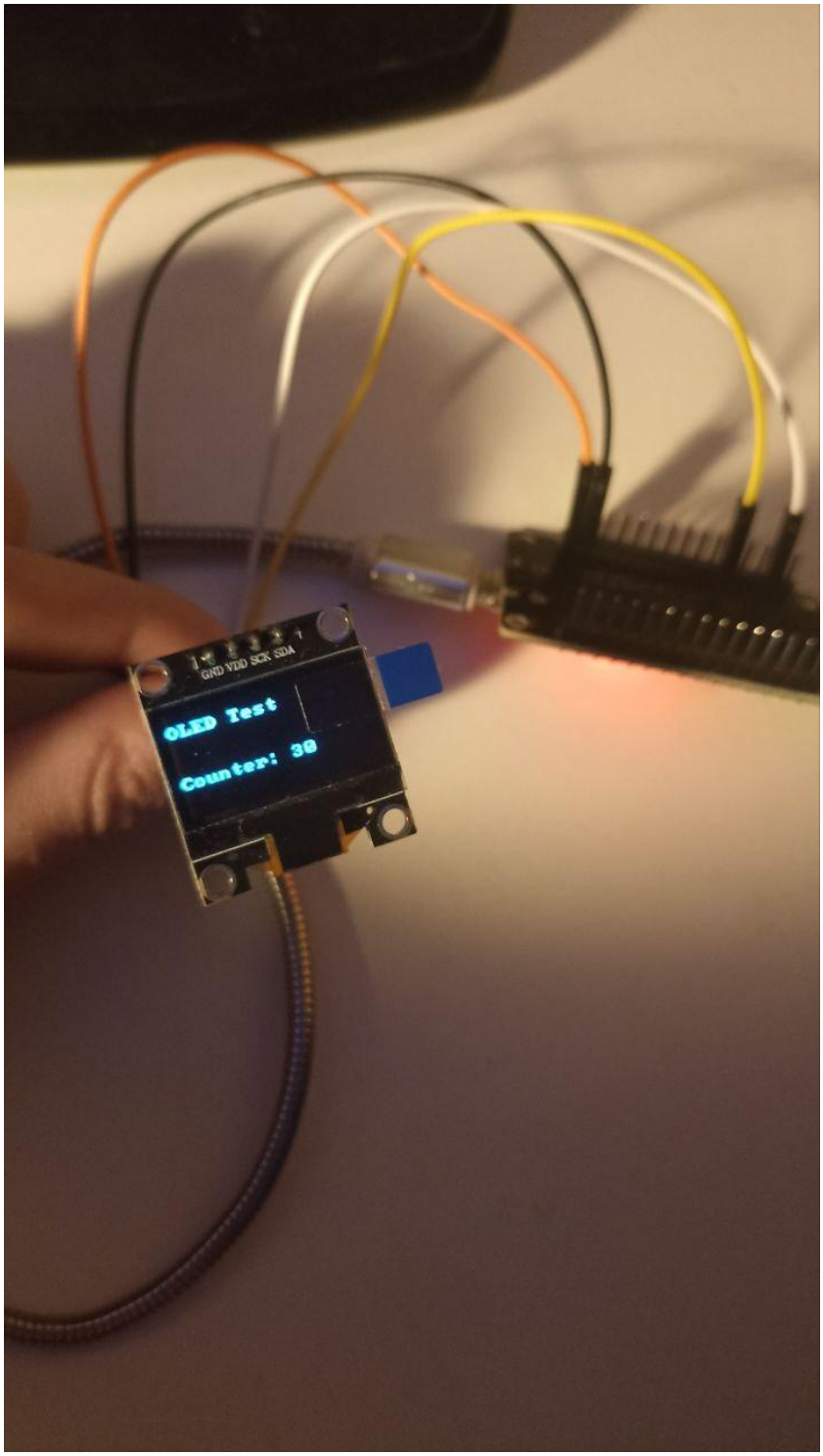
   .clk_flags = 0, *// Флаги тактирования (обычно 0)*

};

Необходимые параметры для config. Существует обратная зависимость: чем выше частота, тем короче может быть шина.

2. Практическая часть:

1. Соберите схему, используя модуль BMP280. Напишите код для получения данных температуры и давления с датчика используя esp-idf, библиотеку для работы с датчиком по вашему выбору. Возможна реализация как в wokwi так и на макетной плате.



```c
#include <stdio.h>
#include <string.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/i2c.h"
#include "ssd1306.h"
#include "font8x8_basic.h"

#define I2C_MASTER_SCL_IO           22
#define I2C_MASTER_SDA_IO           21
#define I2C_MASTER_NUM              I2C_NUM_0
#define I2C_MASTER_FREQ_HZ          400000

void app_main(void) {
    printf("Starting OLED Demo\n");

    SSD1306_t dev;

    i2c_master_init(&dev, I2C_MASTER_SDA_IO, I2C_MASTER_SCL_IO, -1);

    ssd1306_init(&dev, 128, 64);
    printf("OLED initialized\n");

    ssd1306_clear_screen(&dev, false);

    ssd1306_display_text(&dev, 1, "OLED Test", 9, false);

    ssd1306_show_buffer(&dev);
    printf("Text displayed\n");

    int counter = 0;
    while(1) {
        char buffer[20];
        snprintf(buffer, sizeof(buffer), "Counter: %d", counter++);

        ssd1306_display_text(&dev, 5, buffer, strlen(buffer), false);
        ssd1306_show_buffer(&dev);

        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}
```