

А.А. БОНДАРЬ, С.В. ЛИТВИНОВ

**ЦИФРОВЫЕ УСТРОЙСТВА И
МИКРОПРОЦЕССОРЫ**
Семестр 2

Лабораторный практикум
для студентов, обучающихся по направлениям
11.03.01, 11.03.02, 11.03.03, специальности 11.05.01

УДК 004.31 (075.8)

ББК 32.973.26-018

Б73

Бондарь А.А. Цифровые устройства и микропроцессоры. Семестр 2
[Электронный ресурс]: Лабораторный практикум / А.А. Бондарь, С.В. Литвинов. — М.: МИРЭА – Российский технологический университет, 2023. — 1 электрон. опт. диск (CD-ROM).

Лабораторный практикум содержит описание 3-х лабораторных работ, в которых изучаются аппаратные и программные средства микроконтроллеров ESP32, ESP8266.

Материал предназначен для студентов очной формы обучения по направлениям: 11.03.01 «Радиотехника», 11.03.02 «Инфокоммуникационные технологии и системы связи», 11.03.03 «Конструирование и технология электронных средств» и специальности 11.05.01 «Радиоэлектронные системы и комплексы».

Материал может быть использован при изучении дисциплин «Цифровые устройства и микропроцессоры» студентами как очной, так и очно-заочной форм обучения, а также для самостоятельной работы при освоении базового курса кафедры.

Лабораторный практикум издается в авторской редакции.

Авторский коллектив: Бондарь Александр Александрович, Литвинов Святослав Викторович

Рецензенты:

Буханец Дмитрий Иванович, д.т.н., ученый секретарь АО «Радиотехнический институт имени академика А.Л. Минца»

Овчинникова Елена Викторовна, д.т.н., профессор кафедры «Радиофизика, антенны и микроволновая техника», «Московский авиационный институт (национальный исследовательский университет)»

Минимальные системные требования:

Наличие операционной системы Windows, поддерживаемой производителем.

Наличие свободного места в оперативной памяти не менее 128 Мб.

Наличие свободного места в памяти хранения (на жестком диске) не менее 30 Мб.

Наличие интерфейса ввода информации.

Дополнительные программные средства: программа для чтения pdf-файлов (AdobeReader).

Подписано к использованию по решению Редакционно-издательского совета МИРЭА – Российского технологического университета от ____ 2023 г.

Объем 2 Мб

Тираж 10

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
ЛАБОРАТОРНАЯ РАБОТА № 1	6
Общее знакомство с микроконтроллером ESP32 и средой разработки программного обеспечения	6
Лабораторный макет	6
Среда разработки	7
Загрузка и выполнение рабочей программы	9
Порядок выполнения работы	9
1. Ознакомление со средой разработки, операторами языка Си, типами данных	9
2. Построение графиков функций стандартной библиотеки Си	11
3. Изучение цифровых входов/выходов	13
4. Изучение аналоговых входов/выходов	15
5*. Генерация широтно-импульсно модулированного сигнала	17
Содержание отчета	17
Контрольные вопросы	18
ЛАБОРАТОРНАЯ РАБОТА № 2	20
Прерывания, таймеры, символьный и графический дисплей	20
1. Внешние прерывания	20
2. Таймеры	20
3. Символьный дисплей	22
4. Графический дисплей	26
5. Шаговый двигатель*	27
Контрольные вопросы	31
ЛАБОРАТОРНАЯ РАБОТА № 3	33
Интерфейсы UART, I2C	33
1. Интерфейс UART	33
2. Интерфейс I2C	38
3. Wi-fi точка доступа и веб-сервер на ESP32*	43
Контрольные вопросы	45

ДОПОЛНИТЕЛЬНЫЕ СПРАВОЧНЫЕ СВЕДЕНИЯ.....	46
SimulIDE.....	46
Микроконтроллер ESP32.....	49
Микроконтроллер ESP8266.....	50
Микроконтроллер ArduinoUno/Nano	51
Символьный дисплей.....	52
Графический дисплей	53
Таймеры.....	55
СПИСОК ИСТОЧНИКОВ И ЛИТЕРАТУРЫ	58

ВВЕДЕНИЕ

Микроконтроллеры ESP способны выполнять широкий спектр разнообразных технических задач. Это опрос систем различных датчиков, управление устройствами вывода данных, организация локальной беспроводной сети передачи данных, управление двигателями и т.д. Использование микроконтроллеров ESP находит широкое применение в промышленности.

При этом программирование данных микроконтроллеров возможно в простой среде ArduinoIDE, которая легка в освоении и требует минимальный уровень знаний по программированию.

В отличие от плат семейства Arduino, которые также легко программируются, микроконтроллеры ESP обладают лучшими техническими характеристиками (большая тактовая частота, наличие двух ядер, встроенный цифро-аналоговый преобразователь, встроенный модуль Wi-fi), что позволяет использовать их как для обучения, так и для создания сложных промышленных проектов.

В данном практикуме содержатся лабораторные работы, целью которых является дать студентам общее представление об основных принципах работы микроконтроллеров и базовые навыки работы с ними.

ЛАБОРАТОРНАЯ РАБОТА № 1

Общее знакомство с микроконтроллером ESP32 и средой разработки программного обеспечения

Целью лабораторной работы является ознакомление: с основными характеристиками микроконтроллера ESP, со средой разработки программного обеспечения, с форматами и диапазонами обрабатываемых чисел, с особенностями выполнения операций на языке Си, с формами представления выходной информации и выводом её на дисплей. Работа производится на отладочной плате.

Лабораторный макет

На рисунке 1.1 представлен внешний вид отладочной платы и её распиновка.

ESP32 ESP32S 30P

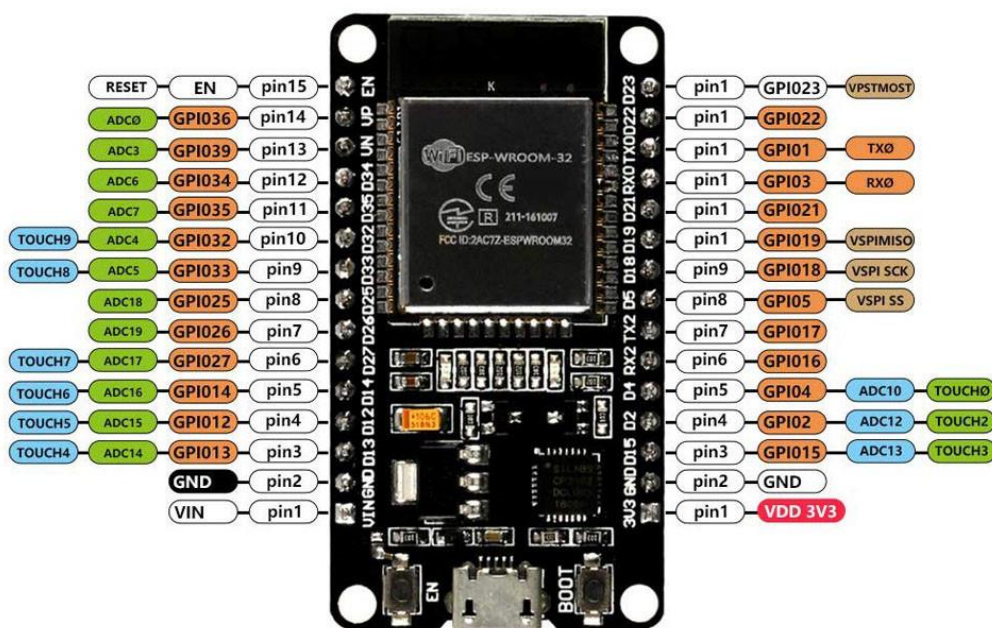


Рисунок 1.1. Распиновка платы микроконтроллера ESP32

На рисунке 1.2 приведена функциональная схема макета.

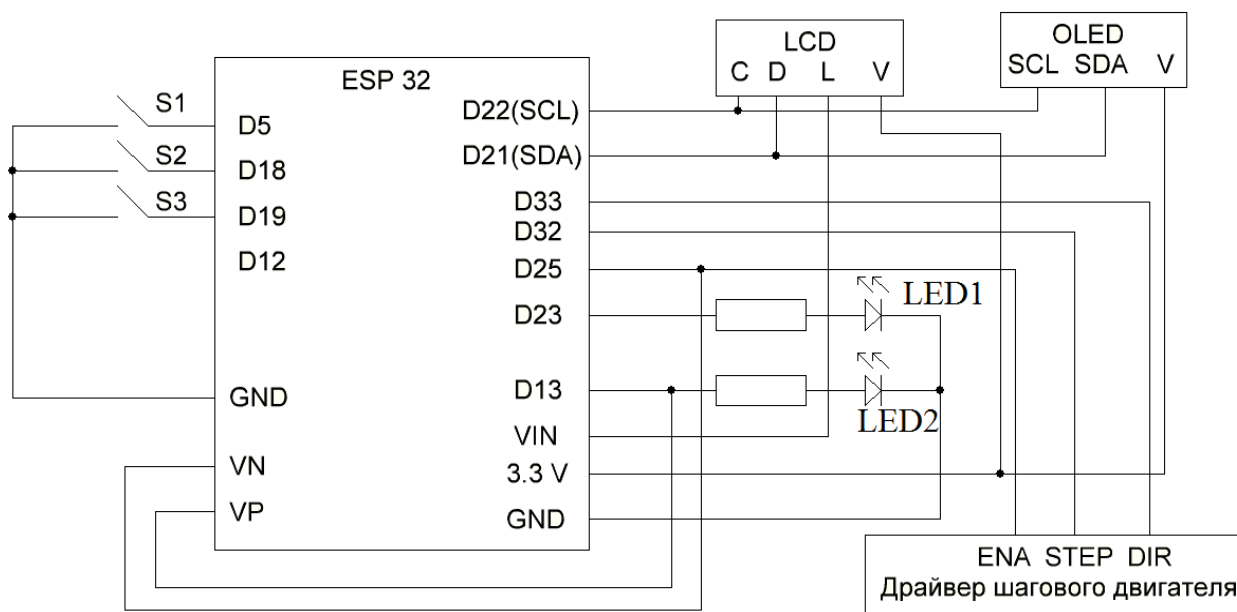


Рисунок 1.2. Функциональная схема лабораторного макета

Схема состоит из микроконтроллера ESP32 и подключенных к нему устройств ввода данных (кнопки S1, S2, S3), устройств вывода данных (светодиоды LED1, LED2, символьный (LCD) и графический (OLED) дисплеи) и шагового двигателя. При необходимости возможно подключение дополнительных устройств (клавиатуры, сервомоторов, двигателей постоянного тока, цифровых и аналоговых датчиков).

Среда разработки

Для разработки программ используется среда ArduinoIDE — интегрированная среда разработки для Windows, MacOS и Linux, разработанная на Си и C++, предназначенная для создания и загрузки программ на Arduino-совместимые платы, а также на платы других производителей. Программный пакет имеет в составе базу данных о микроконтроллерах, которая по мере необходимости может дополняться новыми моделями, библиотеки функций, компилятор на языках C/C++. Пользовательский интерфейс представлен на рисунке 1.3.

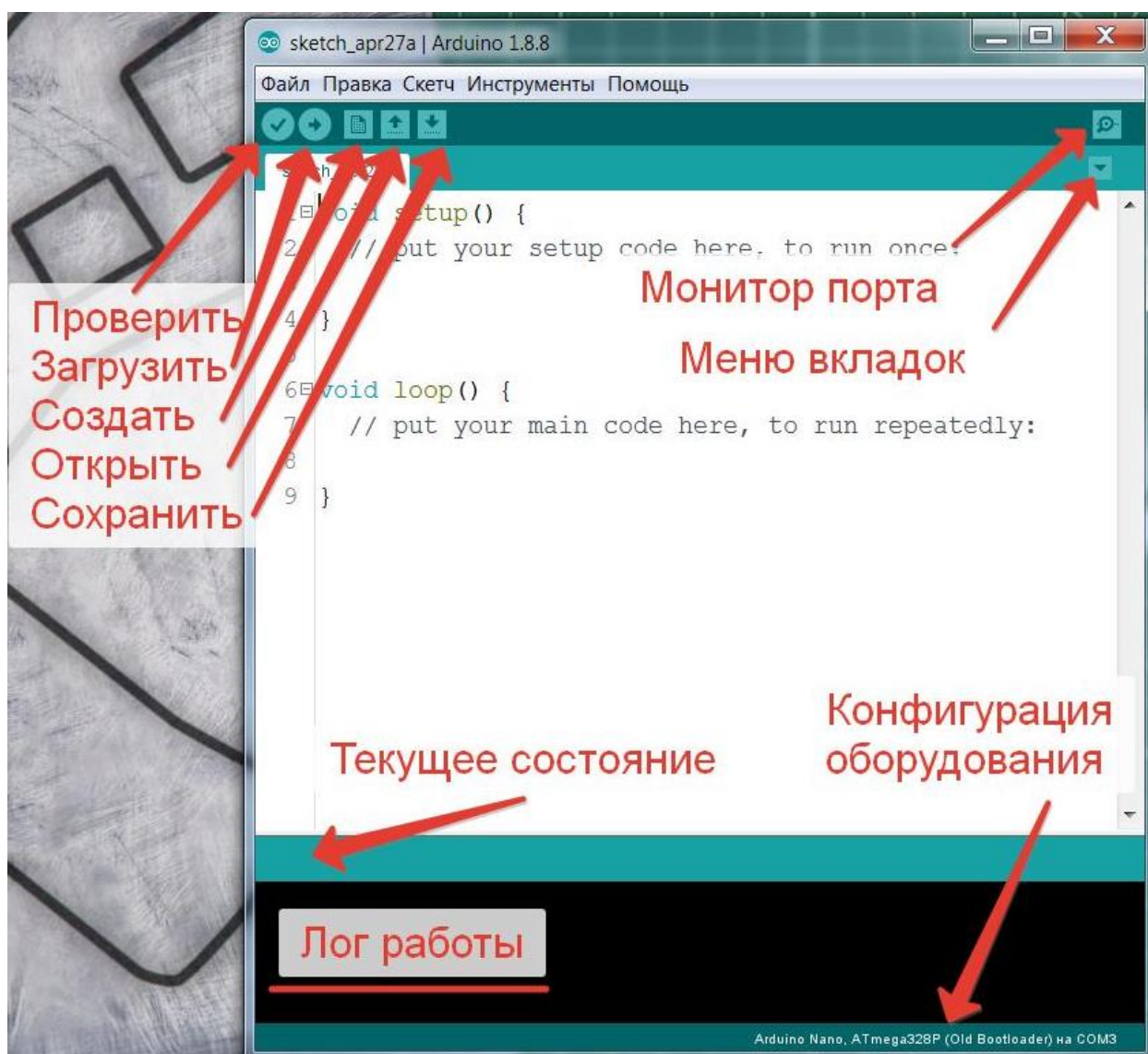



Рисунок 1.3. Интерфейс Arduino IDE

Таблица 1.1. Основные действия при работе в среде Arduino IDE:

Операция	Кнопка
Компилировать проект	
Загрузить проект в плату	
Создать проект	
Открыть	
Закреть	
Открыть монитор порта	
Меню вкладок	

Загрузка и выполнение рабочей программы

Для загрузки программы в плату необходимо подключить микроконтроллер к USB-разъёму компьютера и указать во вкладке «Инструменты → порт» номер USB-порта, к которому подключена плата. Также во вкладке «Инструменты → плата» надо выбрать название используемой платы. Номер порта и модель платы должны появиться в строке «Конфигурация оборудования».

Далее необходимо нажать на кнопку «Загрузить» , после чего начнётся компиляция проекта с последующей загрузкой в плату. В случае с ESP32 при загрузке нужно нажать кнопку «BOOT» на плате и удерживать до окончания загрузки. После завершения загрузки при нажатии на плате на кнопку «Enable», микроконтроллер начнёт выполнение записанной программы с начала.

Кнопка «Enable» выполняет функцию сброса, поэтому при зависаниях программу можно перезапустить нажатием этой кнопки.

Порядок выполнения работы

1. Ознакомление со средой разработки, операторами языка Си, типами данных

Запустить Arduino IDE. На вкладке «Файл» выбрать «Сохранить как» и сохранить текущий, пока пустой проект.

При создании нового файла проекта в Arduino IDE необходимо помнить, что сохраняться он должен в папке, которая имеет название такое же, как и файл. Если производить сохранение в среде Arduino IDE, то такая папка создаётся автоматически.

Убедиться, что в строке конфигурации оборудования отображаются корректные данные, если нет – произвести настройку конфигурации (см. пункт «Среда разработки»).

Загрузить программу «LAB_1_1», которая выдаёт значения синусоиды вида:

$$S(t) = A \cdot \sin(t+d),$$

где A – амплитуда сигнала (изначально принять $A=1$),

t – текущее значение переменной времени,

d – шаг по временной оси (в программе обозначен как shag).

Листинг программы LAB_1_1

```
int16_t res; //целочисленная шестнадцатиразрядная //переменная, благодаря  
ей будем наблюдать эффект //переполнения  
float A=0.8; // задаем амплитуду сигнала
```

```

float shag=0.05; // задаем шаг (период отсчёта)
float t0 = 0; // задаём начальное значение по оси // времени
int ssat(float x, int over); // функция насыщения x – // значение на входе, over –
                             // разрядность

```

```

void setup() {
  Serial.begin(4800); // задаем скорость обмена // данными с портом
}

```

```

void loop() {
  for(int t=0; t<1000; t++){
    res=A*sin(t0+shag*t)*32767; // вычисляем значение // нашей функции в цикле
    // и масштабируем на весь // числовой диапазон
    Serial.println(res);
  }
  delay(1); // даем задержку, чтобы значения функции на // плотере выводились
  // корректно
}

```

```

int ssat(float x, int over){
  int K = 0;
  float predel = pow(2, over)/2;
  if(x > predel-1){
    K = predel-1;
  }
  if(x < -1*predel){
    K = -1*predel;
  }
  if((x > -1*predel) && (x < predel-1))
  {
    K = x;
  }
  return (K);
}

```

Запустить программу на выполнение, убедиться, что код выполняется корректно, открыв плоттер последовательного порта (вкладка «инструменты» →

«плоттер по последовательному соединению»). Зафиксировать в отчёте график на плоттере.

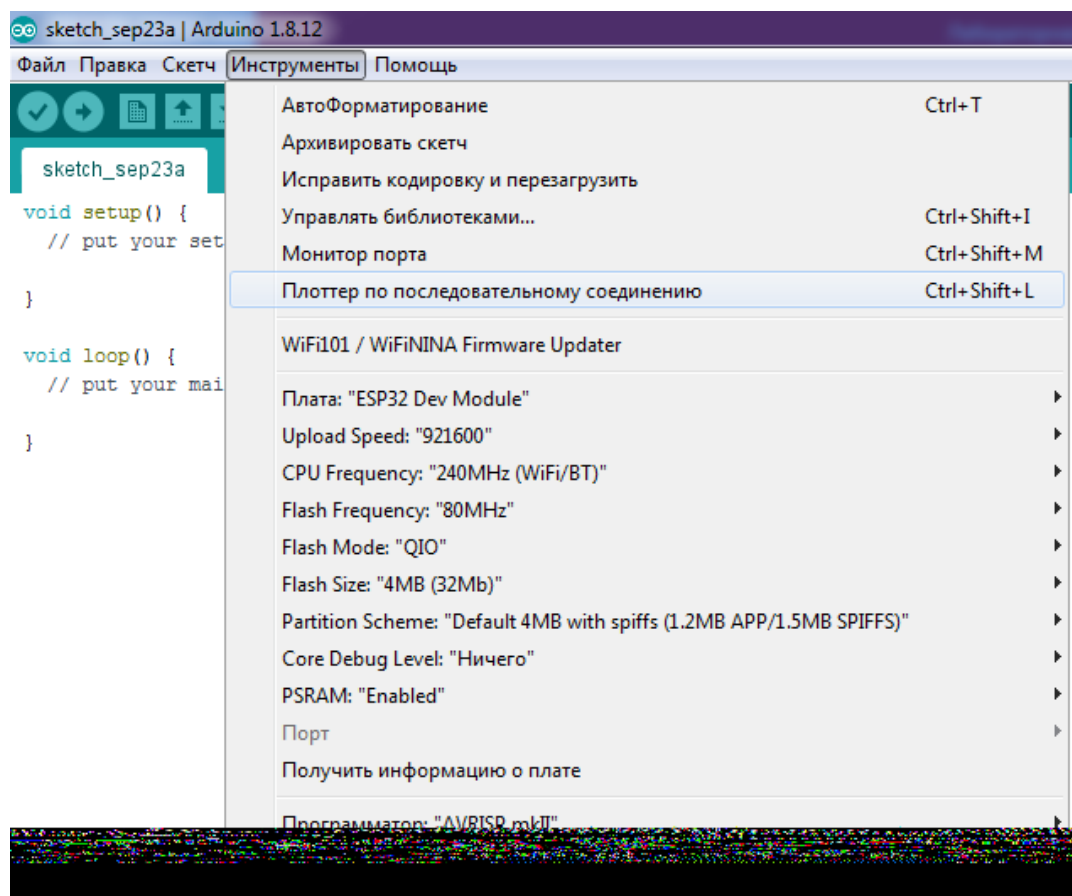


Рисунок 1.4. Выбор режима плоттера

Изменить значение переменной «А» так, чтобы оно стало больше 1, загрузить новый код, открыть плоттер и зафиксировать график сигнала. При защите быть готовым объяснить полученный результат.

Используя функцию `ssat(float входной аргумент, int разрядность)`, компенсировать ошибки при значении амплитуды больше 1 (режим насыщения). В отчёте привести фрагмент кода с функцией и график сигнала на плоттере после применения функции.

2. Построение графиков функций стандартной библиотеки *Си*

Используя пример программы из предыдущего пункта, осуществить построение графика функции по одному из следующих заданий (номер варианта соответствует номеру рабочего места или согласуется с преподавателем). Не использовать насыщение. Если функция в каких-то точках стремится к бесконечности, ограничить (или обнулить) ее значения в некоторой окрестности аргумента. Привести в отчете скорректированные фрагменты программы, скриншоты плоттера.

Таблица 1.2. Варианты заданий:

Вариант	
2.1	Обратная экспонента e^{-x} . Диапазон изменения аргумента $-1 \dots 2$.
2.2	Секанс (выразить через \cos). Диапазон изменения аргумента $-\pi \dots \pi$, особые точки $-\pi/2, \pi/2$
2.3	\exp – экспонента. Диапазон изменения аргумента $-3 \dots 3$.
2.4	Функция $\sin(x)/x$. Диапазон изменения аргумента $-10 \dots 10$. Реализовать корректное поведение вблизи 0.
2.5	\tan – тангенс. Диапазон изменения аргумента $-\pi \dots \pi$, особые точки $-\pi/2, \pi/2$
2.6	\cosh – гиперболический косинус. Диапазон изменения аргумента $-5 \dots 5$.
2.7	Котангенс (выразить через \sin, \cos). Диапазон изменения аргумента $-3\pi/4 \dots +3\pi/4$, особая точка: 0.
2.8	$\text{fmod}(x, 2)$ – дробная часть. Диапазон изменения аргумента $-10 \dots 10$.
2.9	\log_{10} – десятичный логарифм. Диапазон изменения аргумента $0 \dots 10$, особая точка: 0.
2.10	Косеканс (выразить через \sin). Диапазон изменения аргумента $-3\pi/4 \dots +3\pi/4$, особая точка: 0.
2.11	acosh – ареакосинус. Диапазон изменения аргумента $0 \dots 10$. Функция определена для аргумента > 1 .
2.12	Пилообразный сигнал с периодом 3, максимальное значение 10, минимальное 0.
2.13	\sin^3 – кубический синус. Построить 2-3 периода.
2.14	cbrt – кубический корень. Диапазон изменения аргумента $-100 \dots 1000$.
2.15	\log_b – показатель аргумента. Диапазон изменения аргумента $-10 \dots 10$, особая точка: 0.

3. Изучение цифровых входов/выходов

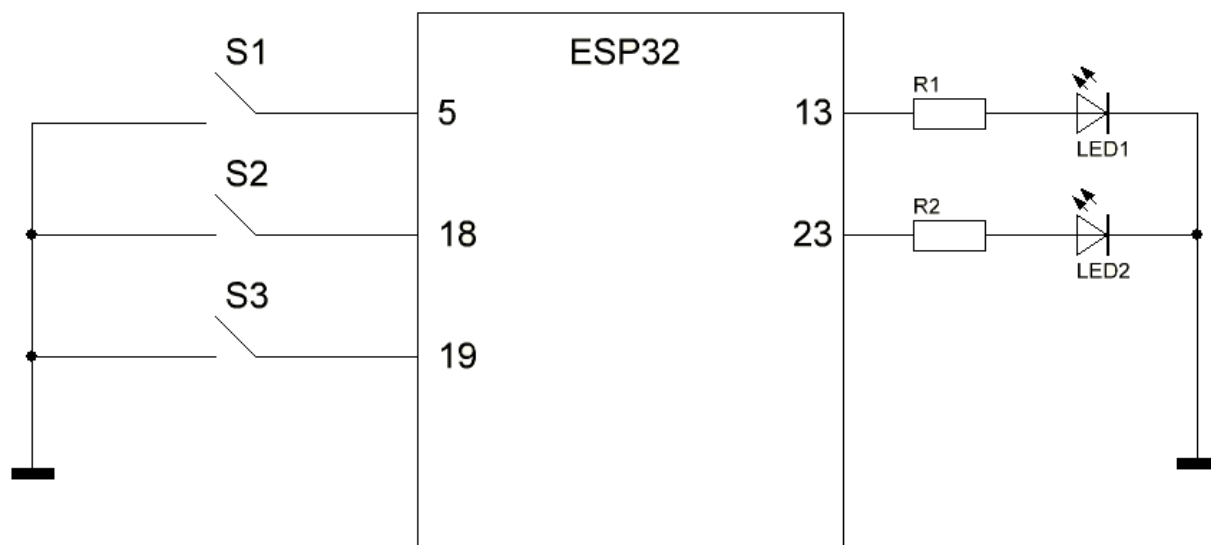


Рисунок 1.5. Схема подключения кнопок и светодиодов в макете

Ознакомиться со схемой макета. Все кнопки подключены к земле и цифровым пинам, настроенным как входы, программно-подтянутые к питанию.

Открыть в Arduino IDE файл-скетч «DIGITAL_IN_OUT». В файле находится скетч программы, управляющей включением и выключением светодиодов LED1 и LED2 с помощью кнопок S1 и S2. Загрузить программу в плату и, разобравшись в принципах её работы, дописать код, подключающий к плате кнопку S3, которая будет управлять светодиодами в соответствии с вариантом задания.

В отчёте привести текст программы.

Таблица 1.3. Варианты заданий:

Вариант	Задание
1	При нажатии S3 светодиоды LED1 и LED2 одновременно загораются и горят пока S3 нажата
2	Пока нажата S3, светодиоды LED1 и LED2 синхронно мигают, каждые 0.5 с изменяя своё состояние
3	Пока нажата S3, светодиоды LED1 и LED2 поочерёдно мигают, каждую секунду изменяя своё состояние
4	При нажатии S3 светодиоды LED1 и LED2 одновременно однократно загораются на 10 с затем гаснут
5	При нажатии S3 поочерёдно загораются сначала LED1 на одну секунду, потом LED2 на одну секунду
6	Пока нажата S3, два раза с длительностью 1 с и паузой 0.5 с мигает LED1, затем один раз с длительностью 0.5 с мигает LED2
7	Пока нажата S3, два раза с длительностью 0.5 с и паузой 0.5 с мигает LED1, затем два раза с длительностью 1 с и паузой 1 с мигает LED2
8	При нажатии S3 поочерёдно выводятся все возможные комбинации на LED1 и LED2 (оба погашены, горит LED1, горит LED2, горят оба). Каждая комбинация выводится в течение одной секунды
9	При нажатии S3 LED1 загорается сразу на 2 с, а LED2 загорается спустя 1 с после нажатия и горит 2 с
10	Пока нажаты S3 и S1, LED1 мигает с периодом в 3 с
11	Пока нажаты S3 и S2, LED2 мигает с периодом в 4 с
12	Пока нажата S3, LED1 и LED2 мигают синхронно каждый раз увеличивая период в два раза (1 с, 2 с, 4с, 8 с). Если период оказывается больше 8 с, то происходит сброс на начальное значение 1 с
13	Пока нажата S3, два раза с длительностью 2 с и паузой 1 с мигает LED2, затем один раз с длительностью 3 с мигает LED1
14	При нажатии S3 светодиоды LED1 и LED2 одновременно дважды мигают с длительностью 2 с и паузой 1 с, затем гаснут
15	При первом S3 светодиоды LED1 и LED2 одновременно загораются и горят, при втором нажатии S3 LED1 и LED2 гаснут

4. Изучение аналоговых входов/выходов

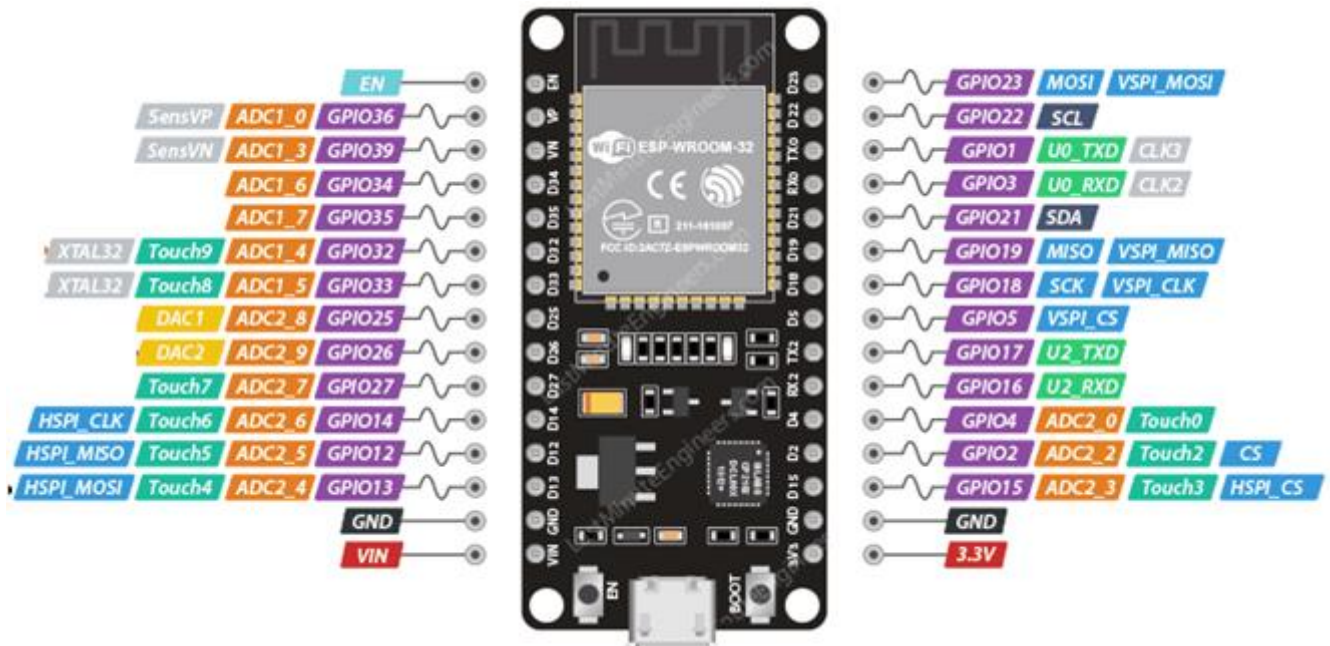


Рисунок 1.6. Распиновка ESP32

В ESP32 есть два пина, подключенные к 8-ми разрядному ЦАП (DAC). Это D25 и D26. Диапазон выдаваемого ЦАП напряжения: от 0 до 3.3 В. Также в ESP32 имеются 15 пинов, подключенных к 12-ти разрядному АЦП (ADC). Диапазон измеряемого АЦП напряжения также находится в пределах от 0 до 3.3 В.

Соединить проводами пины 25 и 36 (на плате обозначен как VP).

Открыть файл «ANALOG_IN_OUT», загрузить программу. В программе циклически с шагом 64 меняется значение кода, подаваемого на ЦАП. В качестве выхода ЦАП указан 25 пин. Также в программе присутствует алгоритм считывания аналогового сигнала с помощью АЦП, подключенного к 36 пину. Значения кодов, отправляющихся на ЦАП и возвращаемых АЦП выводятся на символьный дисплей.

Модифицировать программу так, чтобы переключение кода ЦАП происходило с шагом 32 по нажатию кнопки. Добавить в программу вычисление напряжения по значению кода АЦП. Напряжение выводить на дисплей как третий параметр вместе с кодами ЦАП и АЦП.

Заполнить таблицу 1.4, произведя теоретический расчёт и практические измерения значений кода АЦП и напряжения.

Таблица 1.4. Расчётные и измеренные показания АЦП

Код ЦАП	Код АЦП рас- чёт	$U_{\text{расч}}, \text{В}$	Код АЦП измеренный	$U_{\text{изм}}, \text{В}$
0	0	0	0	0
32				
64				
96				
128				
160				
192				
224				
255	4095	3.3		

Наличие внутренних помех, нестабильностей при аналого-цифровом преобразовании, плохого качества опорного напряжения, внешних флуктуаций во входном сигнале и других факторов обуславливают нестабильность цифрового кода — так называемый шум младших разрядов. Оценить шум можно по величине изменения кода, произведя несколько измерений сигнала на одном из средних значений напряжения (например, при коде ЦАП 128). Зафиксированные коды с АЦП заносятся в таблицу 1.5, после чего находится разность между максимальным и минимальным показателем АЦП. Далее эта разность переводится в двоичный код. Количество двоичных разрядов, занимаемых разностью и будет количеством шумовых разрядов.

Произведя 5 измерений заполнить таблицу 1.5 и определить количество шумовых разрядов.

Таблица 1.5. Влияние шумов на работу АЦП

	Код АЦП
Измерение 1	
Измерение 2	
Измерение 3	
Измерение 4	
Измерение 5	

В отчёте привести заполненные таблицы 1.4 и 1.5, пример расчёта кода АЦП и напряжения, а также расчёт количества шумовых разрядов АЦП, текст модифицированной программы.

5*. Генерация широтно-импульсно модулированного сигнала

Поскольку в микроконтроллерах выходов с ЦАП довольно мало, а порой они вообще отсутствуют, то в качестве альтернативы аналогового сигнала используют ШИМ сигнал. Широтно-импульсная модуляция (ШИМ, англ. pulse-width modulation (PWM)) — процесс управления мощностью методом пульсирующего включения и выключения потребителя энергии.

Выходы, способные выдавать ШИМ сигнал, обозначены на рисунке 1.6 волнистыми линиями. Для исследования ШИМ будем использовать 13 пин как выходной и 39 как вход (на плате обозначен как VN), подключенный к АЦП.

Открыть файл «PWM», загрузить программу в микроконтроллер. Код программы с помощью ШИМ плавно включает и гасит светодиод.

Добавить в программу вывод на плоттер последовательного соединения значения, считываемые с 39 входа. Если отдельные импульсы ШИМ сигнала не различимы, то понизить частоту ШИМ сигнала в 10, либо в 100 раз. При этом уже можно будет различить мигания светодиода. Зафиксировать график ШИМ сигнала.

Для сравнения ШИМ и аналогового сигналов добавить в код управление уровнем напряжения на 25 пине с помощью ЦАП (см пункт 4). Напряжение должно плавно возрастать (код ЦАП меняется с 0 до 255) и плавно уменьшаться. Вывести на плоттер сигнал, приходящий от 25 пина на 36 пин, подключенный к АЦП. Сохранить график аналогового сигнала.

В отчёте привести графики ШИМ сигнала и сигнала с ЦАП, текст модифицированной программы. При защите уметь объяснить их разницу и области применения. Допускается один график, на котором изображены оба сигнала.

Содержание отчета

Таблицы, диаграммы/скриншоты с заголовками и объяснениями возникающих особенностей (указаны в соответствующих пунктах выполнения работы).

Полный исходный текст программы с внесенными коррекциями, коррекции рекомендуется выделить, например, другим шрифтом или вручную.

Текст основного модуля и результаты работы программы по индивидуальному заданию.

Контрольные вопросы

1. Каковы причины нарушения «плавности» изменения сигналов при увеличении их амплитуды?
2. Перечислите основные характеристики используемой в работе платы.
3. Перечислите основные условные и циклические операторы языка C++.
4. Каково назначение плоттера по последовательному соединению?
5. В каком разделе программы прописываются действия, выполняемые однократно при запуске программы?
6. Назовите основные типы данных в языке C++.
7. На примере функции `ssat` из первого пункта объясните, как происходит объявление функции.
8. Каковы причины возникновения некорректных результатов при арифметических операциях? Какие рекомендации по их исправлению?
9. Дайте определение основным характеристикам процессора.
10. В чем измеряется скорость обмена данными по последовательному порту?
11. Чем отличается локальная переменная от глобальной?
12. Какую функцию выполняет компилятор?
13. В чём состоит отличие микроконтроллера от микропроцессора?
14. Какие виды памяти есть у микроконтроллера, используемого в работе?
15. Что такое цифровые порты ввода/вывода, какие режимы работы у них есть?
16. Для чего нужно подтягивать входы либо к питанию, либо к земле?
17. Дайте характеристику основных параметров ЦАП, используемого в микроконтроллере ESP32.
18. Дайте характеристику основных параметров АЦП, используемого в микроконтроллере ESP32.
19. Каким образом ЦАП преобразует цифровой код в аналоговый сигнал?
20. Что такое шумящие разряды, каковы причины их появления?
21. Почему в микроконтроллерах редко используется ЦАП? Какие есть альтернативы?
22. Перечислите основные параметры ШИМ сигнала, задаваемые при его программировании.
23. К чему может привести слишком малая частота широтно-импульсной модуляции при управлении яркостью светодиода?
24. Напишите фрагмент кода программы, опрашивающий состояние пина, подключенного к АЦП и выводящий код с АЦП в монитор последовательного порта.

25. Чем можно объяснить неточность при измерении напряжения с помощью АЦП?
26. Какие виды АЦП вы знаете?

ЛАБОРАТОРНАЯ РАБОТА № 2

Прерывания, таймеры, символьный и графический дисплей

Целью лабораторной работы является ознакомление: с принципами отображения информации на символьном и графическом дисплеях, с использованием внутренних и внешних прерываний, с принципами работы таймеров. Работа производится на стенде.

1. Внешние прерывания

Внешние прерывания срабатывают как отклик на какое-то внешнее аппаратное событие, например, изменение состояния контактов ввода/вывода (GPIO) микроконтроллера.

Модуль ESP32 может обрабатывать до 32 прерываний на каждое ядро.

Для этого необходимо прикрепить (attach) один из доступных цифровых контактов к этому прерыванию и назначить для этого прерывания функцию его обработки (ISR) с помощью функции attachInterrupt.

Работать с прерываниями могут все контакты модуля ESP32. Модуль ESP32 поддерживает 5 типов следующих прерываний: LOW, HIGH, CHANGE, FALLING и RISING.

Загрузить программу LAB_2_2_sin. Программа осуществляет вывод синусоиды в монитор последовательного порта. При нажатии на кнопку S1 синусоида меняется на постоянный нулевой сигнал. В программе нажатие на кнопку проверяется обычным условием без использования прерываний.

Зафиксировать и привести в отчёте снимок плоттера в момент нажатия кнопки S1.

Загрузить программу LAB_2_2_interrupt. Программа также выводит синусоиду в монитор последовательного порта, но, в отличие от предыдущей программы, считывание состояния кнопки S1 происходит по прерыванию.

Зафиксировать и привести в отчёте снимок плоттера в момент нажатия кнопки S1. При защите уметь объяснить разницу снимков плоттера с использованием прерывания и без.

2. Таймеры

Таймер позволяет микроконтроллеру отмерять заданные промежутки времени. Таймер работает параллельно с выполняемыми программами, что позволяет выполнять счёт точно без появления задержек.

В микроконтроллерах ESP32 есть 4 независимых таймера.

Упрощённая структурная схема таймера представлена на рисунке 2.1

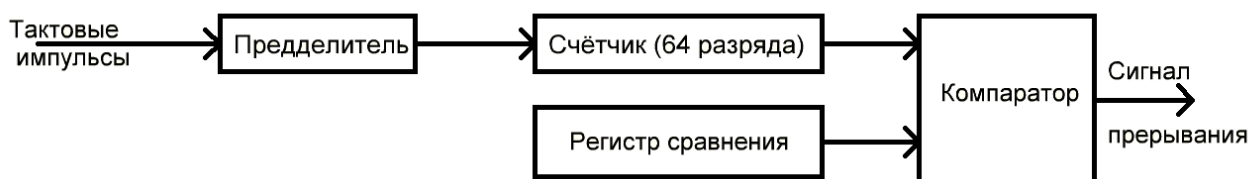


Рисунок 2.1. Структурная схема таймера

На таймер поступают тактовые импульсы, которые являются элементарными единицами отсчёта времени. Поскольку тактовые импульсы имеют частоту 80 МГц, то счётчик будет быстро переполняться при их счёте и таймер не сможет отсчитывать большие промежутки времени (миллисекунды, секунды и т.д.). Для понижения частоты следования тактовых импульсов используется предделитель – устройство, которое понижает частоту тактовых импульсов в заданное количество раз. Импульсы с пониженной частотой поступают на счётчик (в микроконтроллере ESP32 он имеет разрядность 64), который изменяет своё состояние с приходом каждого импульса. Счётчик может быть как суммирующим, так и вычитающим – его параметры задаются в программе. В регистр сравнения записывается значение, до которого должен досчитать счётчик. Сравнение текущего значения счётчика и заданного в регистре осуществляется компаратором, который при совпадении этих значений выдаёт сигнал прерывания.

Открыть программу «LAB_2_2», которая по срабатыванию таймера меняет значение логической переменной, выводимое в плоттер. Запустить программу и, убедившись в корректности её работы, добавить в неё ещё два таймера, которые будут управлять состояниями двух логических переменных, значения которых также выводить в плоттер. Временные интервалы для таймеров задать в соответствии с вариантом в таблице 2.1.

Таблица 2.1. Варианты заданий

Вариант	Таймер 1	Таймер 2
1	20 мс	50 мс
2	80 мс	40 мс
3	60 мс	30 мс
4	100 мс	120 мс
5	90 мс	60 мс
6	50 мс	70 мс
7	30 мс	140 мс
8	130 мс	90 мс
9	15 мс	95 мс
10	25 мс	75 мс
11	65 мс	35 мс
12	85 мс	55 мс
13	105 мс	45 мс
14	115 мс	85 мс
15	125 мс	5 мс

3. Символьный дисплей

Символьный дисплей (LCD) подключается к микроконтроллеру по интерфейсу I2C. Схема подключения символьного и графического дисплеев представлена на рисунке 2.2.

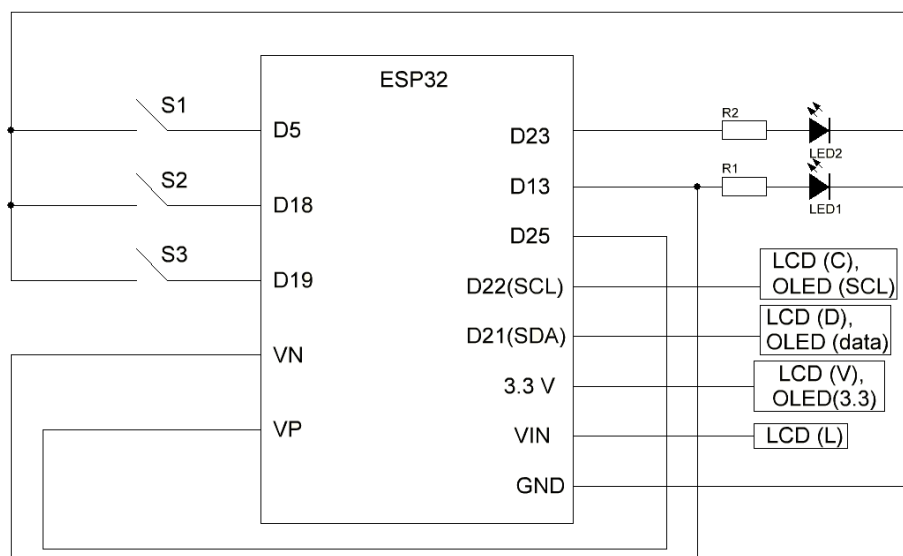


Рисунок 2.2. Схема подключения дисплеев и органов управления

Загрузить программу «LAB_2_3», осуществляющую вывод значений на дисплей, откомпилировать и убедиться в корректности её работы.

Выбрать вариант по номеру рабочего места или получить от преподавателя (при необходимости согласовать также исходные данные).

Для всех параметров (кроме констант) определить переменные, числовые значения задать в программе компактным блоком. Учесть, что при вычислениях необходимо использовать систему единиц СИ. В заданиях с вариацией исходных значений не дублировать расчетную формулу, а включить ее в цикл или в функцию. Осуществить вывод на дисплей исходных данных и конечных результатов. Все вычисления и вывод на дисплей рекомендуется проводить в подпрограмме setup (однократное выполнение). Результаты выполнения программы на дисплее зафиксировать в отчете.

Примечание. В случае отсутствия дисплея у макета в качестве вывода использовать монитор порта. В этом случае код примера выглядит следующим образом:

```
float a = 0.83;
float b = 0.92;

void setup() {
  Serial.begin(9600); // задаём скорость обмена данными с портом
  float w = a + b;
  Serial.println(); //перенос курсора печати на следующую строку
  Serial.print("a+b=");
  Serial.print(w); //вывод значения в порт
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Рисунок 2.2. Скриншот программы для вывода данных в монитор порта

2.1. Рассчитать 3 точки оконной функции Барлетта-Ханна:

$$w(n) = a_0 - a_1 \cdot \left| \frac{n}{N-1} - 0.5 \right| - a_2 \cdot \cos\left(\frac{2\pi \cdot n}{N-1}\right),$$

для $a_0=0.62$; $a_1=0.48$; $a_2=0.38$; $N = 100$; $n = 10, 50$ и 90 .

2.2. Рассчитать резонансную частоту параллельного колебательного контура с потерями:

$$f = \frac{1}{2\pi\sqrt{LC}} \cdot \sqrt{\frac{(L/C)-R^2}{L/C}},$$

при $L = 15$ мкГн, $C = 22$ пФ, $R = 0.1$ кОм и 10 Ом. Вывести значение частоты с единицей измерения без использования степенного множителя в двух форматах: с точностью до 10 знаков после запятой и с округлением до 3-4 знаков.

2.3. Рассчитать полный импеданс последовательной RLC-цепи:

$$Z = \sqrt{R^2 + \left(\omega L - \frac{1}{\omega C}\right)^2},$$

где $\omega = 2\pi f$; $f = 10$ и 50 кГц; $L = 2$ мГн; $C = 330$ нФ; $R = 200$ Ом.

2.4. Определить корни квадратного уравнения

$$0.1234 \cdot x^2 + 5.67 \cdot 10^{-3} \cdot x + 8.9 \cdot 10^{-5} = 0,$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Проверить (программными средствами): если уравнение не имеет вещественных корней, изменить знак одного из коэффициентов, на дисплей вывести также результаты проверки.

2.5. Найти коэффициент передачи активного фильтра верхних частот на двух частотах ω , отстоящих на $\pm 25\%$ от частоты среза: $\omega_0 = 1/RC$:

$$k_\omega = \frac{K \omega^2 R^2 C^2}{\sqrt{(1 - (\omega CR)^2)^2 + (3 - K)^2 (\omega CR)^2}},$$

где $K = 2$ (коэффициент усиления); $C = 22$ нФ; $R = 1.5$ кОм.

2.6. Вычислить значение функции \sin по приведенной ниже приближенной формуле и с использованием библиотечной функции для значения аргумента, соответствующего 30° (перевести в радианы).

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}.$$

2.7. Найти значение автокорреляционной функции для $\tau = 0.5$ мс:

$$\Psi(\tau) = \frac{1}{N} \sum_{i=0}^{N-1} f\left(T \frac{i}{N}\right) \cdot f\left(T \frac{i}{N} - \tau\right),$$

где $f(t) = \sin\left(\frac{2\pi}{T} t\right)$, $T = 1$ мс, $N = 50$.

2.8. Рассчитать емкость двухпроводной линии длиной $l = 1.5$ км, расстояние между проводниками 5 мм, радиус проводников 0.2 и 0.6 мм, диэлектрическая проницаемость вакуума $\epsilon_0 = 8.85 \cdot 10^{-12}$ Ф/м, относительная диэлектрическая проницаемость $\epsilon = 2.5$:

$$C = \frac{2\pi \cdot \epsilon_0 \cdot \epsilon \cdot l}{\ln(d/r)}.$$

2.9. Рассчитать энергию излучения (формула Планка):

$$u_{\lambda T} = \frac{2\pi \cdot h \cdot c^2}{\lambda^5} \cdot \frac{1}{\exp(hc/(k\lambda T)) - 1},$$

где $c = 2.998 \cdot 10^8$ м/с — скорость света; $k = 1.38 \cdot 10^{-23}$ Дж/К — постоянная Больцмана; $h = 6.626 \cdot 10^{-34}$ Дж·с — постоянная Планка; $\lambda = 700$ нм — длина волны (для красного света); $T = 273$ и 373 К — абсолютная температура.

2.10. Рассчитать три точки вольтамперной характеристики р-п перехода для напряжений $u = -0.5, 0$ и $+0.5$ В:

$$i = i_s \cdot \left[\exp \frac{u}{\varphi_T} - 1 \right], \text{ где } \varphi_T = \frac{kT}{q} \text{ — температурный потенциал;}$$

$i_s = 1$ мкА — ток насыщения; $T = 280$ К — абсолютная температура; $k = 1.38 \cdot 10^{-23}$ Дж/К — постоянная Больцмана; $q = 1.6 \cdot 10^{-19}$ Кл — элементарный заряд.

2.11. Найти три точки Гауссовой функции для $\mu = -0.5$, $\sigma^2 = 2$ и вариации $x = -1.5, -0.5, 0.5$:

$$f = \frac{1}{\sigma\sqrt{2\pi}} \exp \left(-\frac{(x-\mu)^2}{2\sigma^2} \right).$$

2.12. Рассчитать три точки Гауссова радиоимпульса:

$$y = \exp(-\alpha t^2) \cdot \cos(2\pi f_c t),$$

где $\alpha = 4.5$ — коэффициент длительности; $f_c = 20$ МГц — частота несущей; $t = -50$ нс, 50 нс, 200 нс — временные отсчеты.

2.13. Вычислить значение экспоненциальной функции, аппроксимированной рядом Маклорена, и той же функции с использованием стандартной библиотеки для аргумента $x = 3.3$:

$$e^x = \sum_{n=0}^3 \frac{x^n}{n!}.$$

2.14. Найти коэффициенты взаимной индуктивности двух антенных катушек с параметрами: числа витков $N_{1,2} = 5$ и 2 ; радиусы витков: $R_{1,2} = 55$ и 33 мм; расстояния между катушками $x = 1, 5, 20$ мм; $\mu_0 = 4\pi \cdot 10^{-6} \text{ В}\cdot\text{с} / \text{А}\cdot\text{м}$ — магнитная постоянная:

$$M = \frac{\mu_0 \cdot N_1 \cdot R_1^2 \cdot N_2 \cdot R_2^2 \cdot \pi}{2 \sqrt{(R_2^2 + x^2)^3}}.$$

2.15. Определить коэффициент ослабления электромагнитной волны при прохождении через слой земли толщиной $d = 5$ и 15 м:

$$k = e^{-pd}, \text{ где } p = \left[\frac{X \cdot B}{2} \cdot \left(\sqrt{1 + \left(\frac{\sigma \cdot 10^9}{B} \right)^2} - 1 \right) \right]^{1/2},$$

$X = 0.8 \cdot 10^{-16} \pi \cdot f$; $B = 0.556 \epsilon \cdot f$; $\sigma = 3 \cdot 10^{-2} \text{ См/м}$ — проводимость земли; $\epsilon = 5$ — диэлектрическая проницаемость земли; $f = 10 \text{ МГц}$ — частота волны.

4. Графический дисплей

Открыть файл «OLED» и загрузить программу на плату. Программа по нажатию кнопки S1 выводит на графический дисплей два периода функции $\sin(x)$, а по нажатию кнопки S2 — два периода функции $\cos(x)$. Изучив программу, доработать её, добавив вывод третьей функции по нажатию кнопки S3. Функция определяется в соответствии с вариантом по таблице 2.2. В отчёте привести текст модифицированной программы, фото дисплея с заданным графиком.

Таблица 2.2. Варианты заданий

Вариант	Функция
1	$\sin^2(x)$, два периода
2	$\cos^2(x)$, два периода
3	$\text{cas}(x) = \sin(x) + \cos(x)$, два периода
4	Прямоугольный периодический сигнал, два периода
5	Пилообразный сигнал, два периода
6	Треугольный сигнал, три периода
7	$\sin^3(x)$, два периода
8	$\cos^3(x)$, два периода
9	Пилообразный сигнал, три периода
10	Треугольный сигнал, два периода

11	Прямоугольный периодический сигнал, пять периодов
12	$\sin^2(x)$, четыре периода
13	$\cos^2(x)$, шесть периодов
14	$\text{cas}(x) = \sin(x) + \cos(x)$, три периода
15	$\cos^3(x)$, три периода

5. Шаговый двигатель*

Шаговый двигатель способен поворачивать свой вал на фиксированный угол (шаг), который определяется количеством полюсов ротора и статора.

Принцип действия простейшего шагового двигателя показан на рисунке 2.3. Магнитный ротор поворачивается при переключении электромагнитов, расположенных на неподвижном статоре. На рисунке 2.3 количество электромагнитов на статоре равно четырём, следовательно, ротор с валом может находиться в одном из четырёх положений. Шаг поворота такого двигателя составляет 90° .

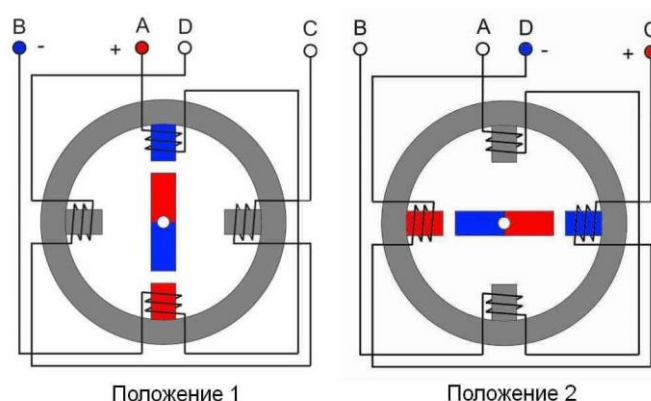


Рисунок 2.3. Принцип работы шагового двигателя

В рассматриваемом примере, все электромагниты попарно соединяются в две обмотки: верхний и нижний – обмотка А-В, правый и левый – обмотка С-Д. В зависимости от направления протекания тока в каждой обмотке на электромагнитах будет возникать магнитное поле определённой полярности, в соответствии с которой будет менять своё положение ротор. Управляя током в обмотках А-В, С-Д, можно управлять положением ротора.

В настоящих шаговых двигателях количество полюсов существенно больше четырёх. Так в двигателе 28BYJ-48, используемом в стенде их 32, то есть, у ротора двигателя есть 32 угловых положения. Кроме того, в двигателе 28BYJ-48 имеется встроенный редуктор, повышающий крутящий момент и имеющий передаточное отношение 1:64. Таким образом, за один оборот внешнего вала ротор

двигателя совершит 64 оборота. Поэтому количество угловых положений внешнего вала равно 2048.

Для управления током в двух обмотках двигателя используется плата-переходник – драйвер двигателя, преобразующий сигналы управления с микроконтроллера в соответствующие токи, поступающие на обмотки двигателя. Поскольку токи, поступающие на обмотки двигателя должны быть большими, и микроконтроллер такие токи выдать не может, то к драйверу двигателя дополнительно подсоединяется силовая линия питания.

Схема подключения шагового двигателя в лабораторном стенде представлена на рисунке 2.4.

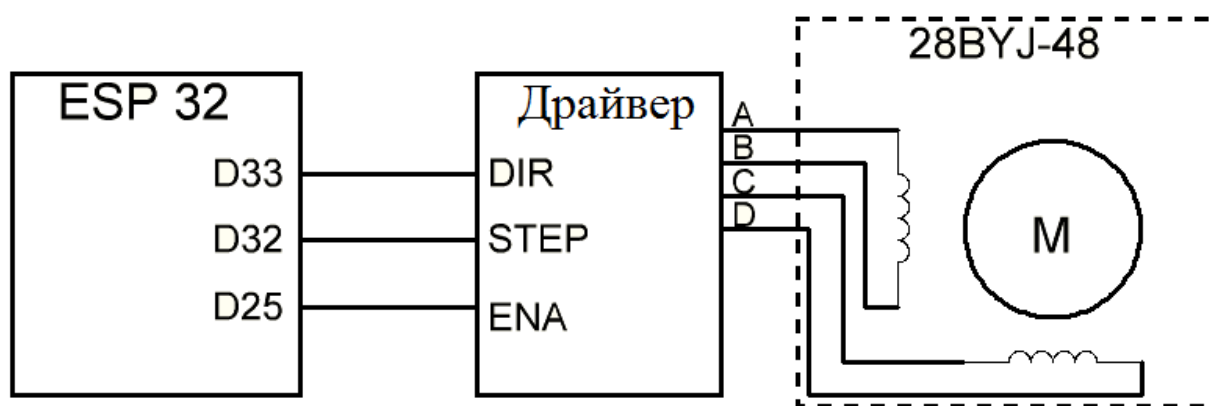


Рисунок 2.4. Схема подключения шагового двигателя к стенду

Как видно из рисунка, обмотки двигателя подключены к драйверу, который управляется тремя сигналами, поступающими от микроконтроллера: DIR – сигнал, отвечающий за направление вращения двигателя (1 – в одну сторону, 0 – в другую), STEP – сигнал, отвечающий за поворот ротора на один шаг (в стенде таким сигналом является восходящий фронт), ENA – сигнал, разрешающий или запрещающий работу драйвера (1 – разрешено, 0 – запрещено). Данные три управляющих сигнала являются универсальными для большинства драйверов шаговых двигателей и позволяют обходиться без дополнительных библиотек.

Подключить шнур питания драйвера стенда к сетевой розетке. Загрузить программу «STEPPER» в микроконтроллер. Программа содержит пример кода, управляющего вращением шагового двигателя. Модифицировать программу в соответствии с вариантом индивидуального задания.

1. Написать программу для управления шаговым двигателем с помощью кнопок S1 – включает/выключает драйвер, S2 – определяет направление, S3 – выбирает одну из двух скоростей.
2. Параметры скорости и направления выводить на символьный дисплей.

3. Написать программу, которая каждые 5 с будет менять направление вращения шагового двигателя, используя для отсчёта времени таймер. Направление вращения выводить на графический дисплей.
4. Написать программу для управления светодиодом LED1 и шаговым двигателем с помощью кнопок. Кнопка S1 отвечает за управление светодиодом (включить/выключить), кнопка S2 отвечает за изменение направления вращения двигателя. Для обработки сигналов с кнопок использовать прерывания. Предусмотреть условие, если светодиод горит, то вращение двигателя запрещено.
5. Написать программу для управления шаговым двигателем с помощью кнопок S1 – включает/выключает драйвер, S2 – определяет направление. Светодиод LED1 горит при вращении двигателя по часовой стрелке, светодиод LED2 – против. Если мотор не вращается, то светодиоды погашены.
6. Написать программу, которая с помощью кнопок S1 и S2 управляет направлением вращения шагового двигателя. Нажатие на кнопку S3 генерирует сигнал прерывания, которое останавливает двигатель.
7. Написать программу, которая каждые 10 с будет менять скорость вращения шагового двигателя, используя для отсчёта времени таймер. Скоростей всего задать 3. Текущую скорость выводить на символьный дисплей.
8. Написать программу, которая с помощью кнопок S1 и S2 управляет направлением вращения шагового двигателя. Светодиод LED1 горит при вращении двигателя по часовой стрелке, светодиод LED2 – против. Если мотор не вращается, то светодиоды погашены. При одновременном нажатии на кнопки S1 и S2 происходит остановка двигателя.

Контрольные вопросы

1. Какие сигналы могут использоваться внешними прерываниями?
2. В чём отличие прерывания от обычного условного оператора?
3. Какие входы микроконтроллера ESP32 поддерживают работу с внешними прерываниями?
4. На примере программ из лабораторной работы объясните, что такое функция-обработчик прерывания, как она объявляется и какими свойствами она должна обладать.
5. Какие функции может выполнять таймер в микроконтроллере? Сколько таймеров есть у ESP32?
6. Приведите схему устройства таймера в микроконтроллере, объясните назначение всех устройств в таймере.
7. На примере программ из лабораторной работы объясните как происходит объявление таймера и задание его параметров.
8. Дайте классификацию прерываний.
9. Какую информацию может выдать символьный дисплей?
10. Какие характеристики есть у графического дисплея, в каком виде он может выдавать информацию?
11. Какие команды используются для вывода текста на графический дисплей?
12. Напишите фрагмент кода программы, задающей отсчёт временного интервала в 15 с с помощью таймера.
13. Какие параметры задаются при программировании символьного дисплея, используемого в лабораторной работе?
14. Какие параметры задаются при программировании графического дисплея, используемого в лабораторной работе?
15. В каких случаях необходимо и оправдано использование прерываний?
16. В чём заключается принцип работы шагового двигателя?
17. Назовите характеристики шагового двигателя, использованного в работе.
18. Какими сигналами осуществляется управление драйвером шагового двигателя? За что отвечает каждый сигнал?
19. Почему рекомендуется отключать не используемый шаговый двигатель от питания?
20. Обязательно ли для управления шаговым двигателем использовать специализированные библиотеки? Какие способы управления использованы в работе?
21. Какие виды таймеров вам известны?
22. Что такое глубина прерываний? Какая глубина прерываний у ESP32?

23. Напишите фрагмент кода программы, заставляющий шаговый двигатель совершить один шаг.

ЛАБОРАТОРНАЯ РАБОТА № 3

Интерфейсы UART, I2C

Целью лабораторной работы является ознакомление: с принципами передачи данных по интерфейсам UART и I2C. Работа производится на стенде.

1. Интерфейс UART

UART – это двунаправленный последовательный асинхронный интерфейс для связи двух устройств. Для UART есть специальные выделенные пины (RX – приёмник, TX – передатчик). Специальных пинов может быть несколько (на каждую пару приёмник-передатчик отведен один интерфейс) как ESP32 или одна пара, как на ESP8266. Схема обозначения пинов приведена на рисунках 3.1 и 3.2. Схема соединения ESP8266 и ESP32 приведена на рисунке 3.3.

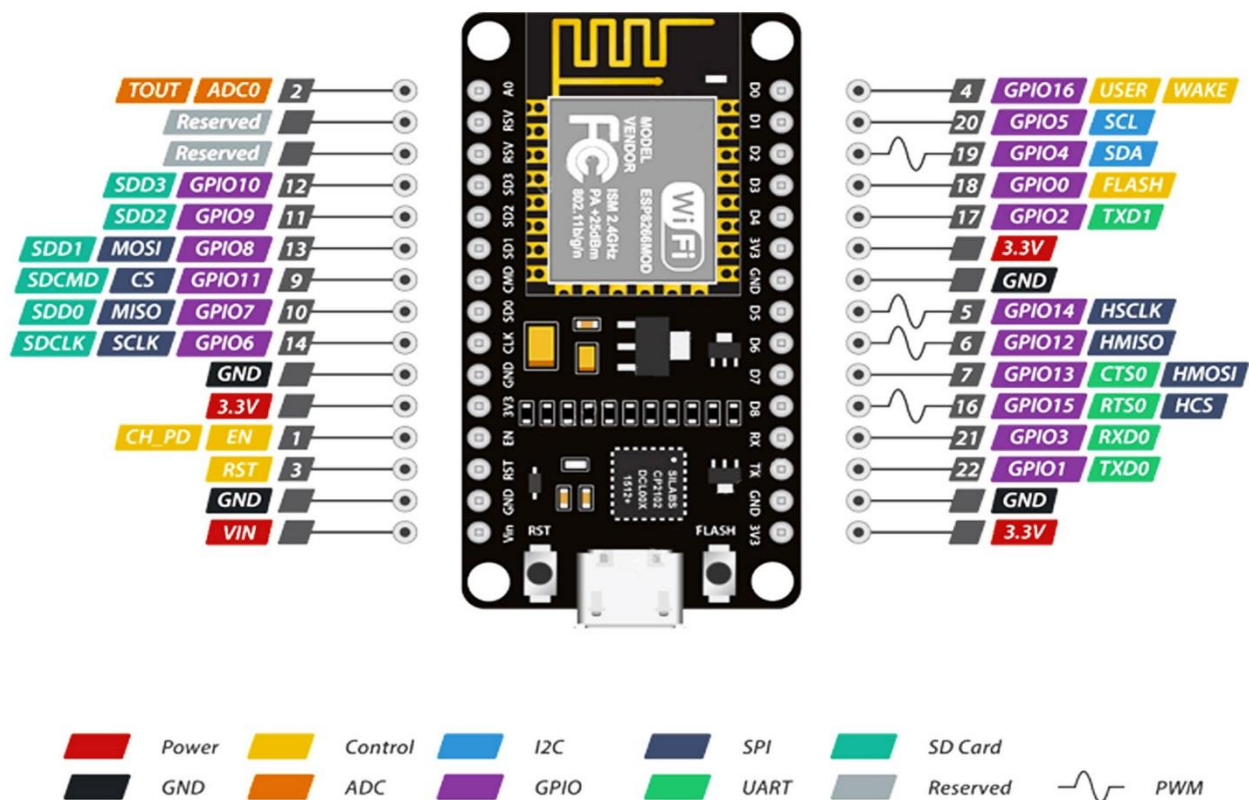


Рисунок 3.1. Пины ESP8266

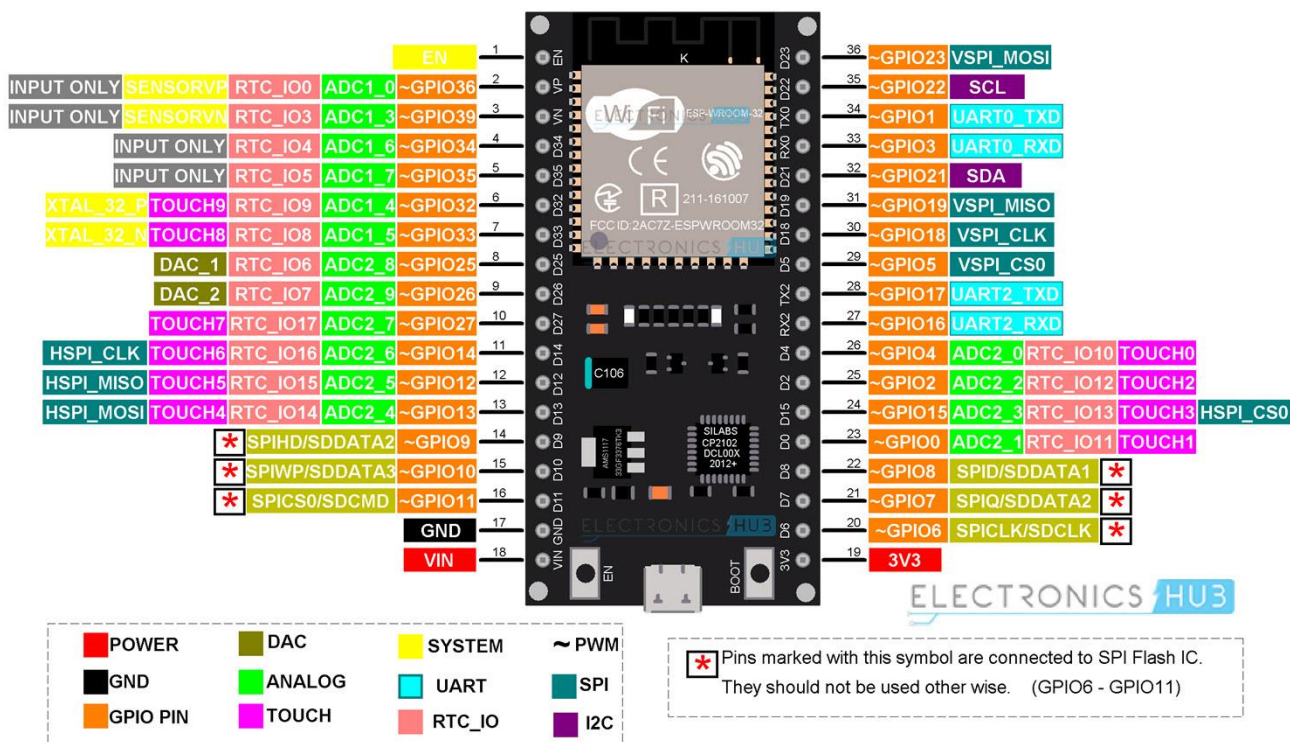


Рисунок 3.2. Пины ESP32

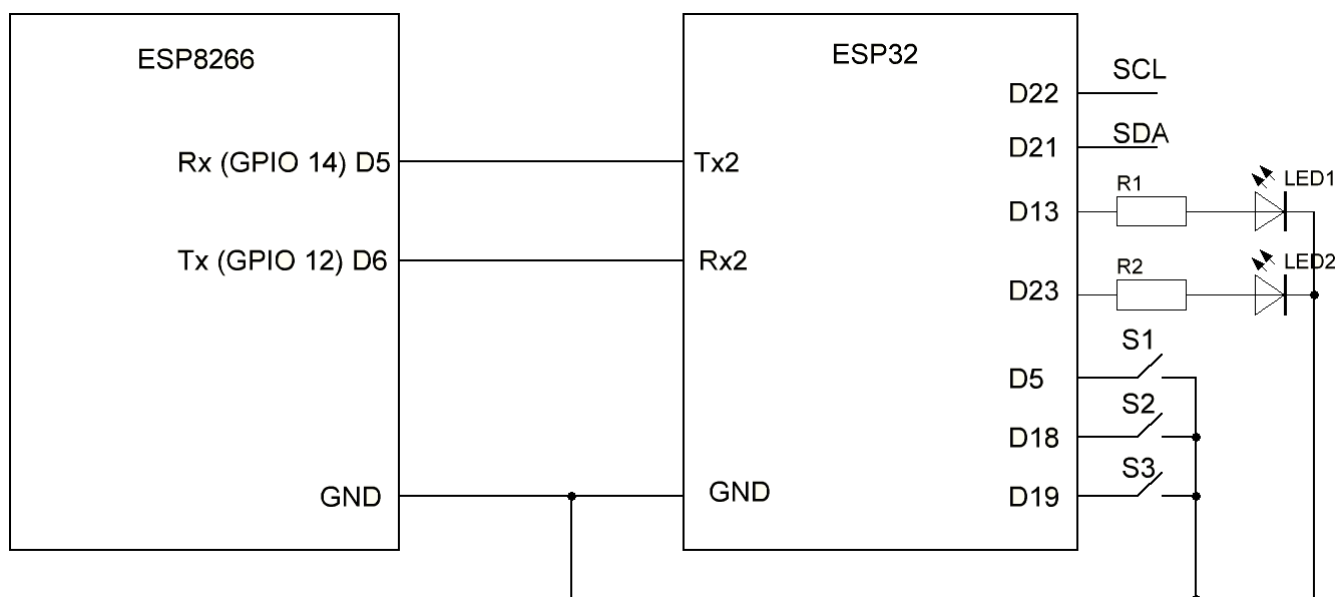


Рисунок 3.3. Соединение МК по UART

Соединить платы в соответствии со схемой на рисунке 3. Поскольку нулевой интерфейс UART (Tx0, Rx0) задействован для программирования плат и обмена данными с компьютером, то необходимо использовать дополнительные пины. На ESP32 это пины второго интерфейса UART (TX2, RX2). Подключение их осуществляется командой «Serial2.begin(9600);», которая запускает второй

интерфейс. Соответственно, если нужно обратиться именно ко второму интерфейсу, то обращение начинается с его названия «Serial2».

У ESP8266 только один встроенный интерфейс UART (Tx, Rx). Через этот интерфейс происходит программирование платы и обмен данными с компьютером. Поэтому, если предполагается общение платы по последовательному порту с компьютером, то эти пины нельзя использовать для связи с другими устройствами. Выходом из ситуации является использование библиотеки «SoftwareSerial.h», которая позволяет подключить практически любые два пина как входы/выходы UART. В нашем случае приёмным пином (Rx) будет назначен пин D5 (GPIO 14), а передающим (Tx) – D6 (GPIO 12).

Открыть в папке лабораторной работы файлы «UART_ESP32» и «UART_ESP8266». Подключить обе платы к компьютеру и загрузить на каждую из них соответствующий код. Перед загрузкой обязательно убедиться, что в загрузчике указан верный COM порт и модель платы.

Программа для ESP8266 считывает сообщение из командной строки последовательного порта (см рисунок 3.4) и отправляет его по UART на ESP32. Также программа принимает данные от платы ESP32 и печатает их в мониторе порта (В примере в качестве данных выступает символ «а»).



Рисунок 3.4. При нажатии Enter или кнопки «Отправить» сообщение из командной строки считается платой.

Программа для ESP32 принимает по UART сообщения от ESP8266 и выводит их на первую строку символьного дисплея. При нажатии на кнопку S1 ESP32 передаёт на ESP8266 символ «а».

Ознакомившись с программами дописать их в соответствии с вариантами задания.

Таблица 3.1. Варианты заданий

1	<p>ESP8266 отправляет сообщения, считываемые из командной строки монитора порта на ESP32. Если сообщением являлось слово «AND», то ESP32 отправляет на ESP8266 таблицу истинности логического элемента «И», которая выводится в мониторе порта ESP8266.</p> <p>Примерный вид таблицы:</p> <p style="margin-left: 40px;">X1 = 0 X2 = 0 Y = 0</p> <p style="margin-left: 40px;">X1 = 0 X2 = 1 Y = 0</p> <p style="margin-left: 40px;">X1 = 1 X2 = 0 Y = 0</p> <p style="margin-left: 40px;">X1 = 1 X2 = 1 Y = 1</p>
2	<p>ESP8266 посылает по интерфейсу UART на ESP32 числа от 0 до 100 в порядке возрастания, каждую секунду увеличивая значение на единицу. При нажатии кнопки S2, ESP32 отправляет ESP8266 по UART команду вести счёт в порядке убывания. При нажатии кнопки S1, ESP32 отправляет ESP8266 по UART команду вести счёт в порядке возрастания.</p>
3	<p>При получении сообщения «LED_1 ON» из командной строки последовательного порта ESP8266 отправляет на ESP32 команду зажечь светодиод LED1. Командой «LED_1 OFF» светодиод гасится.</p>
4	<p>ESP8266 отправляет сообщения, считываемые из командной строки монитора порта на ESP32. Если сообщением являлось слово «OR», то ESP32 отправляет на ESP8266 таблицу истинности логического элемента «ИЛИ», которая выводится в мониторе порта ESP8266.</p> <p>Примерный вид таблицы:</p> <p style="margin-left: 40px;">X1 = 0 X2 = 0 Y = 0</p> <p style="margin-left: 40px;">X1 = 0 X2 = 1 Y = 1</p> <p style="margin-left: 40px;">X1 = 1 X2 = 0 Y = 1</p> <p style="margin-left: 40px;">X1 = 1 X2 = 1 Y = 1</p>
5	<p>ESP8266 посылает по интерфейсу UART на ESP32 число 2, возведенное в степень n (n меняется от 0 до 10, каждую секунду увеличивая значение на единицу). При нажатии кнопки S2, ESP32 отправляет ESP8266 по UART команду вести уменьшать n. При нажатии кнопки S1, ESP32 отправляет ESP8266 по UART команду увеличивать n.</p>

6	При получении сообщения «LED_ON» из командной строки последовательного порта ESP8266 отправляет на ESP32 команду зажечь светодиоды LED1, LED2. Командой «LED_OFF» светодиоды гасятся.
7	ESP8266 отправляет сообщения, считываемые из командной строки монитора порта на ESP32. Если сообщением являлось слово «NAND», то ESP32 отправляет на ESP8266 таблицу истинности логического элемента «И-НЕ», которая выводится в мониторе порта ESP8266. Примерный вид таблицы см. варианты 1, 4.
8	ESP8266 посылает по интерфейсу UART на ESP32 числа от 0 до 100 в порядке возрастания, каждую секунду увеличивая значение на единицу. При нажатии кнопки S2, ESP32 отправляет ESP8266 по UART команду сделать шаг счёта равным двум (каждую секунду число увеличивается на 2). При нажатии кнопки S3, ESP32 отправляет ESP8266 по UART команду сделать шаг счёта равным трём. При нажатии кнопки S1, ESP32 отправляет ESP8266 по UART команду сделать шаг счёта равным одному.
9	При получении сообщения «LED_CHANGE» из командной строки последовательного порта ESP8266 отправляет на ESP32 команду зажигать светодиоды LED1, LED2 попеременно с периодом 1 с. Командой «STOP» светодиоды гасятся.
10	ESP8266 отправляет сообщения, считываемые из командной строки монитора порта на ESP32. Если сообщением являлось слово «NOR», то ESP32 отправляет на ESP8266 таблицу истинности логического элемента «ИЛИ-НЕ», которая выводится в мониторе порта ESP8266. Примерный вид таблицы см. варианты 1, 4.
11	ESP8266 посылает по интерфейсу UART на ESP32 числа от 0 до 100 в порядке возрастания, каждую секунду увеличивая значение на единицу. При нажатии кнопки S2, ESP32 отправляет ESP8266 по UART команду вести счёт в порядке убывания. При нажатии кнопки S1, ESP32 отправляет ESP8266 по UART команду вести счёт в порядке возрастания. При нажатии кнопки S3, ESP32 отправляет ESP8266 по UART команду остановить счёт.
12	При получении сообщения «LED_1 ON» из командной строки последовательного порта ESP8266 отправляет на ESP32 команду зажечь светодиод LED1. Командой «LED_2 ON» зажигается светодиод LED2. Командой «LED_OFF» оба светодиода одновременно гасятся.

13	ESP8266 отправляет сообщения, считываемые из командной строки монитора порта на ESP32. Если сообщением являлось слово «XOR», то ESP32 отправляет на ESP8266 таблицу истинности логического элемента «Исключающее ИЛИ», которая выводится в мониторе порта ESP8266. Примерный вид таблицы см. варианты 1, 4.
14	ESP8266 посылает по интерфейсу UART на ESP32 числа от 0 до 100 в порядке возрастания, каждую секунду увеличивая значение на единицу. При нажатии кнопки S2, ESP32 отправляет ESP8266 по UART команду сделать шаг счёта равным двум (каждую секунду число увеличивается на 2). При нажатии кнопки S1, ESP32 отправляет ESP8266 по UART команду произвести сброс счётчика. При нажатии кнопки S3, ESP32 отправляет ESP8266 по UART команду остановить счёт.
15	При получении сообщения «LED_ON» из командной строки последовательного порта ESP8266 отправляет на ESP32 команду зажечь светодиоды LED1, LED2. Командой «LED_2 OFF» гасится светодиод LED2. Командой «LED_1 OFF» гасится светодиод LED1.

2. Интерфейс I2C

Для работы с интерфейсом I2C используется библиотека «Wire.h».

Основные команды библиотеки:

Wire.begin(address) – производит инициализацию библиотеки Wire и осуществляет подключение к шине I2C в качестве ведущего (master) или ведомого (slave). 7-битный адрес ведомого в данной команде является опциональным и если он не указан [Wire.begin()], то устройство подключается к шине I2C в качестве ведущего (master).

Wire.read() – считывает принятый байт.

Wire.write() – Записывает данные в устройство, являющееся ведомым или ведущим.

Возможны варианты:

Wire.write(value), где value – значение передаваемого одиночного байта;

Wire.write(string) – для передачи последовательности байт.

Wire.write(data, length), где data – массив данных для передачи в виде байт, length – число байт для передачи.

Wire.beginTransmission(address) – начало передачи от ведущего к ведомому. Address – адрес ведомого, к которому обращаются.

Wire.endTransmission() – завершение передачи.

Wire.onRequest() – в скобках указывается функция, которая вызывается при поступлении запроса с главного устройства.

Wire.onReceive() – в скобках указывается функция, которая вызывается в подчинённом при получении сообщения от главного.

Wire.requestFrom(address,quantity) – запрос от главного устройства к подчинённому. Главное требует в качестве ответа выдать количество байт, указанных в параметре quantity. Address – адрес ведомого, к которому обращаются.

Для подключения устройств по шине I2C необходимо три провода (GND, SCLK, SDA). Открыть файл «I2C_signal», загрузить его на плату ESP32, снять с помощью осциллографа диаграмму сигналов SCLK и SDA. В отчёте привести диаграмму сигнала с побитовой расшифровкой.

Соедините платы Arduino UNO и ESP32 как показано на рисунке 3.5. В работе всегда рассматриваем ESP32 как главное устройство (Master). Открыть файлы «I2C_ESP32» и «I2C_UNO», загрузить их на плату ESP32 и Arduino UNO соответственно. Ознакомиться с принципами работы программы.

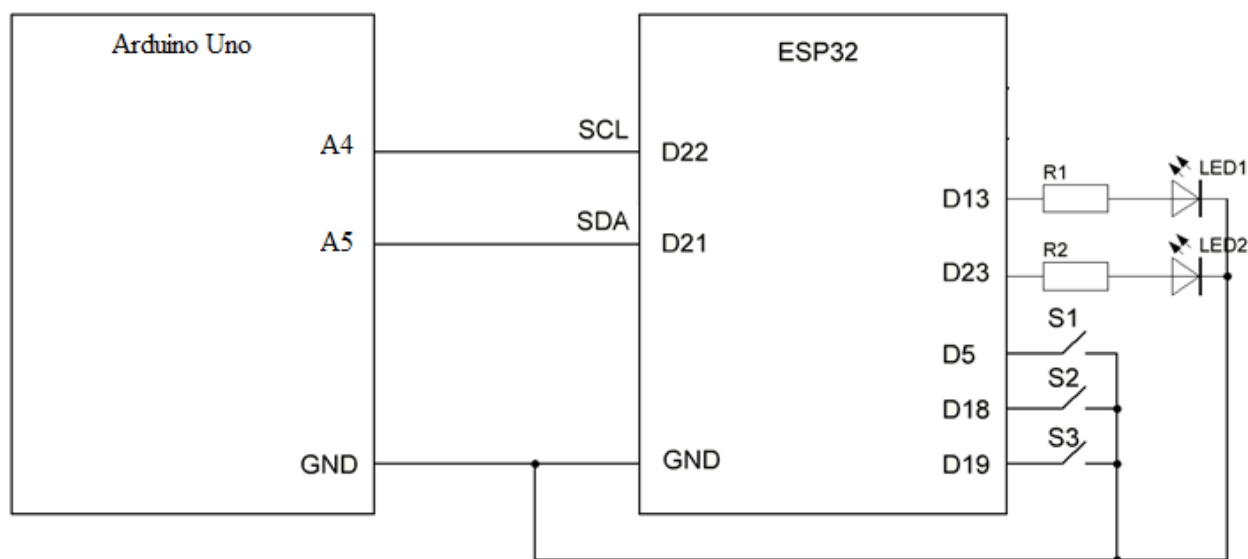


Рисунок 3.5. Соединение МК по I2C

Изучив основы работы с интерфейсом I2C, выполнить идеальные задания в соответствии с вариантом.

Таблица 3.2. Варианты заданий

1	Главное устройство (ESP32) посылает на подчинённое (Arduino UNO) номера нажатых кнопок. Следом за номером кнопки следует запрос на данные с подчинённого. В зависимости от принятого номера подчинённое устройство в качестве ответа на запрос выдаёт сообщения главному «BUT_1_pressed», «BUT_2_pressed», «BUT_3_pressed». Принятые сообщения главное устройство вводит на дисплей.
2	Главное устройство (ESP32) посылает на подчинённое (Arduino UNO) информацию о состоянии светодиодов, управляемых кнопками S1, S2. Подчинённое устройство выводит полученные сообщения в монитор последовательного порта. Примерсообщений: LED_1_ON, LED_1_OFF, LED_2_ON, LED_2_OFF
3	Главное устройство (ESP32) при нажатии на кнопку S1 посылает на подчинённое (Arduino UNO) команду зажечь встроенный светодиод (на плате Arduino UNO подключён к 13 пину). При нажатии на кнопку S2 подаётся команда погасить встроенный светодиод. По запросу от главного подчинённое устройство передаёт сообщение о состоянии светодиода (LED_ON, LED_OFF). Запрос от главного поступает при нажатии кнопки S3. Ответ подчинённого выводится на дисплей.
4	Главное устройство (ESP32) посылает на подчинённое (Arduino UNO) однобайтные сообщения (например «А», «В», «С»). Сообщения выбираются кнопками S1, S2, S3. Следом за сообщением следует запрос на подчинённое устройство. В зависимости от сообщения подчинённое устройство в качестве ответа на запрос выдаёт сообщения главному «Wassend А», «Wassend В», «Wassend С». Принятые сообщения главное устройство вводит на дисплей.
5	Главное устройство (ESP32) посылает на подчинённое (Arduino UNO) информацию о состоянии светодиодов, управляемых кнопками S1, S2. Подчинённое устройство выводит полученные сообщения в монитор последовательного порта. Пример сообщений: LED_1_ON, LED_1_OFF, LED_2_ON, LED_2_OFF Если светодиоды горят или погашены одновременно, то передаётся одно сообщение (LEDS_ON, LEDS_OFF).

6	Главное устройство (ESP32) при нажатии на кнопку S1 посылает на подчинённое (Arduino UNO) команду зажечь встроенный светодиод (на плате Arduino UNO подключён к 13 пину). При нажатии на кнопку S2 подаётся команда погасить встроенный светодиод. При нажатии на кнопку S3 подаётся команда 5 раз мигнуть светодиодом с периодом в 0,5 секунды. По запросу от главного подчинённое устройство передаёт сообщение о состоянии светодиода (LED_ON, LED_OFF). Ответ подчинённого выводится на дисплей.
7	Главное устройство (ESP32) посылает на подчинённое (Arduino UNO) количество нажатий кнопки S1 при каждом новом нажатии. Если количество нажатий было больше 10, то подчинённое устройство зажигает встроенный светодиод (подсоединён к 13 пину).
8	Главное устройство (ESP32) при нажатии на кнопку S1 посылает на подчинённое (Arduino UNO) команду изменить значение логической переменной x1 на противоположное, а при нажатии на кнопку S2 – изменить значение логической переменной x2. При нажатии на кнопку S3 главное устройство посылает запрос подчинённому, по которому подчинённое устройство должно выслать результат логического умножения x1 и x2. Значения переменных x1 и x2 выводятся в монитор порта подчинённого устройства, результат умножения – на дисплей главного устройства.
9	Главное устройство (ESP32) при нажатии на кнопку S1 посылает на подчинённое устройство (Arduino UNO) команду мигнуть встроенным светодиодом (подсоединён к 13 пину) 1 раз (длительность горения светодиода – 0.5 с), при нажатии на кнопку S2 – мигнуть 2 раза, при нажатии на кнопку S3 – мигнуть 3 раза.
10	Главное устройство (ESP32) при нажатии на кнопку S1 посылает на подчинённое (Arduino UNO) команду изменить значение логической переменной x1 на противоположное, а при нажатии на кнопку S2 – изменить значение логической переменной x2. При нажатии на кнопку S3 главное устройство посылает запрос подчинённому, по которому подчинённое устройство должно выслать результат логического сложения x1 и x2. Значения переменных x1 и x2 выводятся в монитор порта подчинённого устройства, результат сложения – на дисплей главного устройства.

11	Главное устройство (ESP32) передаёт подчинённому (Arduino UNO) данные, вводимые через монитор порта главного устройства. Подчинённое устройство выводит все принятые сообщения в монитор порта. Если была введена команда «LED_ON», то подчинённое устройство зажигает встроенный светодиод (подсоединён к 13 пину). Если главное устройство передаёт команду «LED_OFF», подчинённое выключает светодиод.
12	Главное устройство (ESP32) при нажатии на кнопку S1 посылает на подчинённое (Arduino UNO) команду увеличить значение целой переменной x1 на единицу, а при нажатии на кнопку S2 – увеличить значение целой переменной x2 на единицу. Если значение переменных превышает 10, то сбрасывать их в ноль. При нажатии на кнопку S3 главное устройство посылает запрос подчинённому, по которому подчинённое устройство должно выслать результат умножения x1 и x2. Значения переменных x1 и x2 выводятся в монитор порта подчинённого устройства, результат умножения – на дисплей главного устройства.
13	Главное устройство (ESP32) посылает на подчинённое (Arduino UNO) номера нажатых кнопок (1, 2, 3). Следом за номером кнопки следует запрос на данные с подчинённого. Подчинённое устройство в зависимости от принятого числа мигает светодиодом один раз для 1, два раза для 2 и т.д. В качестве ответа на запрос подчинённое выдаёт сообщения главному «BUT_1_pressed», «BUT_2_pressed», «BUT_3_pressed». Принятые сообщения главное устройство вводит на дисплей.
14	Главное устройство (ESP32) при нажатии на кнопку S1 посылает на подчинённое (Arduino UNO) команду увеличить значение целой переменной x1 на единицу, а при нажатии на кнопку S2 – увеличить значение целой переменной x2 на единицу. Если значение переменных превышает 10, то сбрасывать их в ноль. При нажатии на кнопку S3 главное устройство посылает запрос подчинённому, по которому подчинённое устройство должно выслать результат целочисленного деления x1 и x2. Значения переменных x1 и x2 выводятся в монитор порта подчинённого устройства, результат целочисленного деления – на дисплей главного устройства.
15	Главное устройство (ESP32) посылает на подчинённое (Arduino UNO) количество нажатий кнопки S2 при каждом новом нажатии. Если количество нажатий было больше 15, то подчинённое устройство зажигает встроенный светодиод (подсоединён к 13 пину).

3. Wi-fi точка доступа и веб-сервер на ESP32*

Используемый в стенде микроконтроллер ESP32 имеет встроенный Wi-Fi модуль, и способен поддерживать связь по Wi-Fi соединению с другими электронными устройствами. Кроме того, ESP32 может создавать и поддерживать автономный веб-сервер.

Веб-сервер – это место, где хранятся, обрабатываются и отсылаются веб-страницы веб-клиентам. Веб-клиент – это веб-браузер различных гаджетов (телефоны, планшеты, ноутбуки и т.д.). Связь между клиентом и сервером происходит с использованием специального протокола, называемого протоколом передачи гипертекста (HTTP).

ESP32 может не только подключаться к существующей сети Wi-Fi, но и организовывать свою собственную сеть. В этом случае ESP32 выступает в роли точки доступа Wi-Fi сети. В режиме точки доступа ESP32 создает новую сеть Wi-Fi и устанавливает для нее SSID (имя сети) и IP-адрес. С этим IP-адресом ESP32 может отображать веб-страницы на всех подключенных устройствах в своей собственной сети. Максимальное количество устройств, которое можно подключить по Wi-Fi сети к ESP32, равно пяти.

Открыть и изучить программу «Wi-Fi_server». В константу «ssid» записать название для своей Wi-Fi сети, а в константу «password» – пароль, для доступа в сеть. После этого загрузить программу в микроконтроллер. После загрузки программы ESP32 будет работать в режиме точки доступа, к которой можно подключиться с любого гаджета, поддерживающего обмен данными по Wi-Fi. С помощью телефона и компьютера подключитесь к сети, организованной вашим микроконтроллером. Откройте на подключенном устройстве веб-браузер и в адресной строке введите адрес локального веб-сервера, созданного ESP32, 192.168.1.1. На появившейся веб-странице отображаются две кнопки, отвечающие за управление светодиодами, а также надписи, сообщающие о текущем состоянии светодиодов. Убедившись в корректной работе веб-сервера (светодиоды на стенде управляются нажатием кнопок на веб-странице), модифицируйте программу в соответствии с вариантом задания в таблице 3.3.

Таблица 3.3. Варианты заданий

1	Осуществить вывод на веб-страницу состояние трёх кнопок S1, S2, S3. Для каждой кнопки на веб-странице выделить отдельную область с текстом (Нажата/Отпущена).
2	Организовать управление шаговым двигателем с веб-страницы. На странице расположить кнопки «ВПЕРЕД», «НАЗАД», отвечающие за направление вращения двигателя. «СТОП» – за остановку, «ПУСК» – за начало работы, «Скорость +» – за увеличение скорости, «Скорость -» – за уменьшение скорости. Скоростей предусмотреть не меньше трёх.
3	Организовать управление яркостью свечения светодиодов с веб-страницы. Для каждого из двух светодиодов на странице должны быть кнопки: «ВЫКЛ», «25%», «50%», «75%», «МАХ», соответствующие уровням яркости.
4	На веб-странице разместить клавиши: «Прямоугольник», «Круг», «Треугольник». При нажатии на клавишу соответствующая фигура должна выводиться на графическом дисплее, подключённом к ESP32.
5	С помощью веб-страницы (клавиш «X1», «X2») менять значения логических переменных X1 и X2. Текущие значения переменных и их логическую сумму отображать на символьном дисплее.
6	С помощью кнопок на веб-странице («sin», «cos», «sin ² », «sin ³ ») управлять выводом графиков соответствующих сигналов в плоттере монитора порта или на графическом дисплее.
7	Добавить к управлению светодиодами с веб-страницы управление шаговым двигателем (клавиши «вперед», «назад», «старт», «стоп»). Текущее состояние двигателя отображать на веб-странице.
8	С помощью веб-страницы (клавиш «X1», «X2») менять значения логических переменных X1 и X2. Текущие значения переменных и их логическое произведение отображать на символьном дисплее.
9	Выводить на веб-страницу показания с АЦП (см лабораторная 1 пункт 4), которое обрабатывает сигнал, приходящий с ЦАП. Уровень сигнала на ЦАП задаётся кнопками на веб-странице «+5» и «-5».
10	На веб-странице разместить клавиши: «И», «ИЛИ», «НЕ». При нажатии на клавишу соответствующий выбранной логической операции элемент должен выводиться на графическом дисплее, подключённом к ESP32.

Контрольные вопросы

1. Какие соединения (входы/выходы) используются для подключения устройств по интерфейсу UART, для чего они служат?
2. В чём измеряется скорость передачи данных по интерфейсу UART, какой она может быть?
3. Почему для подключения нескольких устройств по интерфейсу UART нельзя использовать одни и те же пины?
4. Изобразите диаграмму сигнала UART интерфейса, объясните её состав.
5. Изобразите схему подключения нескольких (трёх) устройств по интерфейсу I2C, обозначьте все линии связи, объясните принцип работы соединения.
6. Нарисуйте диаграммы сигналов интерфейса I2C для обращения к подчинённому устройству с адресом 7.
7. Какое максимальное количество устройств можно подключить к одной шине I2C? Почему?
8. Перечислите все устройства в учебном стенде, соединённые по интерфейсу I2C.
9. Сколько аппаратных входов/выходов интерфейса UART есть у ESP32?
10. Проведите сравнительную оценку интерфейсов UART и I2C, рассмотрев их характеристики. Дайте рекомендации в каких случаях какой из рассмотренных интерфейсов лучше использовать.
11. В чём отличие интерфейсов UART и RS323? Для сравнения изобразите диаграммы сигналов этих интерфейсов.
12. Какова разница между симметричным и несимметричным интерфейсами? Какое преимущество есть у симметричных интерфейсов?
13. Дайте классификацию интерфейсов передачи данных.
14. Перечислите основные характеристики интерфейса передачи данных, использованных в лабораторной работе.
15. Какие меры по контролю за целостностью передачи данных используются в рассмотренных интерфейсах?

ДОПОЛНИТЕЛЬНЫЕ СПРАВОЧНЫЕ СВЕДЕНИЯ

SimulIDE

SimulIDE – это простой симулятор, позволяющий моделировать основные процессы, происходящие при работе микроконтроллера. Такой симулятор способен выступить неполноценным заменителем лабораторного стенда. SimulIDE при работе не требует доступа к интернету, а также содержит минимальный набор дополнительных компонентов, позволяющих изучить работу микроконтроллера с периферийными устройствами. Вместо ESP32 в SimulIDE будет использоваться микроконтроллер ArduinoUno или Arduino nano, которые по ряду характеристик уступают ESP32, однако по принципам программирования и работы весьма близки к лабораторному микроконтроллеру.

Важный момент: у среды SimulIDE нет своего компилятора, поэтому все написанные коды компилируются в ArduinoIDE, а в симуляторе указывается путь к компилятору. SimulIDE корректно работает с версией ArduinoIDE 1.8.12. При работе с более поздними версиями симулятор может выдавать ошибку.

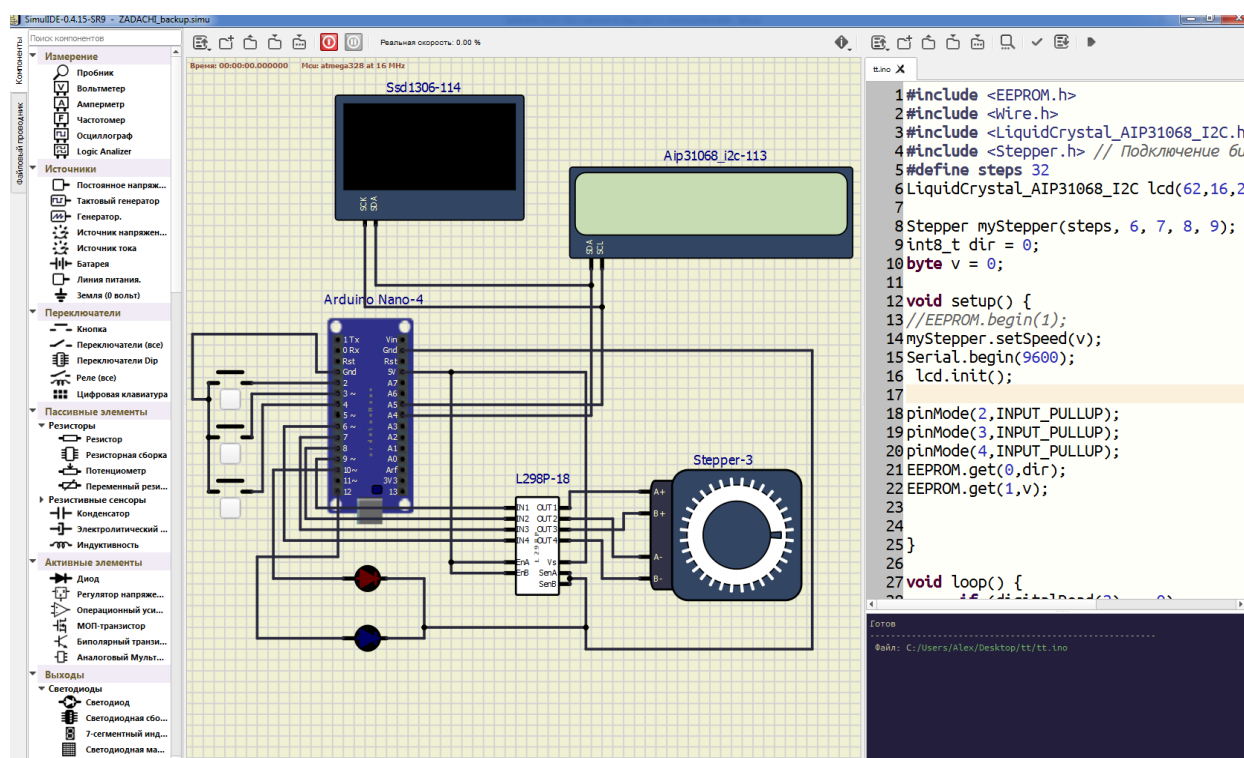


Рисунок 4.1. Общий вид рабочего окна SimulIDE

На рисунке 4.1 приведён общий вид рабочего окна среды SimulIDE. В левой колонке расположена вкладка компонентов, которые можно добавлять в схему. Центральное окно – сама схема, в общих чертах повторяющая схему лаборатор-

ного стенда: к микроконтроллеру Arduino nano подключены три кнопки, два светодиода, символьный и графический дисплеи, шаговый двигатель. В правом окне располагается текст программы, загружаемой в микроконтроллер.

Файл программы создаётся в среде Arduino IDE и открывается с помощью соответствующего значка в окне SimulIDE.



Рисунок 4.2. Значок открытия файла в SimulIDE

После открытия файла с текстом программы необходимо при первом запуске симулятора указать путь к компилятору. Для этого нужно нажать правой кнопкой мыши на вкладку с названием файла кода программы, затем в появившемся меню выбрать пункт «задать путь к компилятору», после чего указать путь до папки Arduino.

Прежде чем загружать код в микроконтроллер его необходимо откомпилировать. Для этого надо нажать на соответствующую кнопку (см. рисунок 4.3).

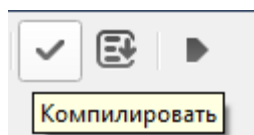


Рисунок 4.3. Компиляция файла в SimulIDE

В нижнем правом углу рабочего окна программы (см. рисунок 4.1) находится окно, отображающее состояние кода. В нём должно появиться сообщение об успешной компиляции. Если компиляция была успешно завершена, то для загрузки в плату необходимо нажать значок «загрузить», расположенный справа от значка «компилировать».

После загрузки программы, нажав на кнопку «запустить симуляцию», можно наблюдать за моделированием работы микроконтроллера.

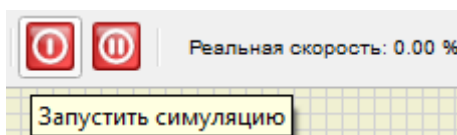


Рисунок 4.4. Запуск симуляции в SimulIDE

Среда SimulIDE содержит множество дополнительных устройств, позволяющих наблюдать за работой схемы (осциллограф, вольтметр, амперметр, логический анализатор и т.д.).

Поскольку повторить в точности все компоненты стенда в симуляторе невозможно, то ниже приведены аналоги, имеющиеся в симуляторе.

В SimulIDE нет символьного дисплея, используемого в стенде. Альтернативой является символьный дисплей Aip31068_I2C, который работает с библиотекой «LiquidCrystal_AIP31068_I2C».

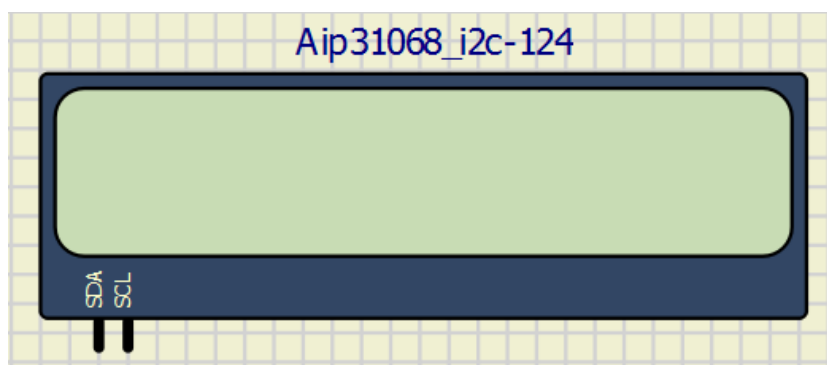


Рисунок 4.5. Символьный дисплей, используемый в симуляторе

Дисплей соединяется с микроконтроллером по интерфейсу I2C и объявляется с помощью следующих команд:

LiquidCrystal_AIP31068_I2C lcd(62,16,2); – объявить объект жидкокристаллический дисплей типа AIP31068_I2C. Имя объекта – lcd (по аналогии с названием переменной). В скобках первое число – адрес устройства в шине I2C (по умолчанию 62), второе число – количество столбцов, третье – количество строк.

В подпрограмме установки (setup) дисплей с именем lcd инициализируется командой «**lcd.init();**». Команды по выводу данных на дисплей такие же, как и для дисплея, используемого на стенде (см ниже пункт «Символьный дисплей»).

В симуляторе имеется драйвер шагового двигателя L298P, который довольно сильно отличается от драйвера, используемого в стенде. L298P предназначен для работы с шаговым биполярным двигателем, поэтому при построении схемы в симуляторе в свойствах шагового двигателя необходимо поставить значение true в параметре «биполярный». Возможная схема подключения драйвера и шагового двигателя к микроконтроллеру показана на рисунке 4.6.

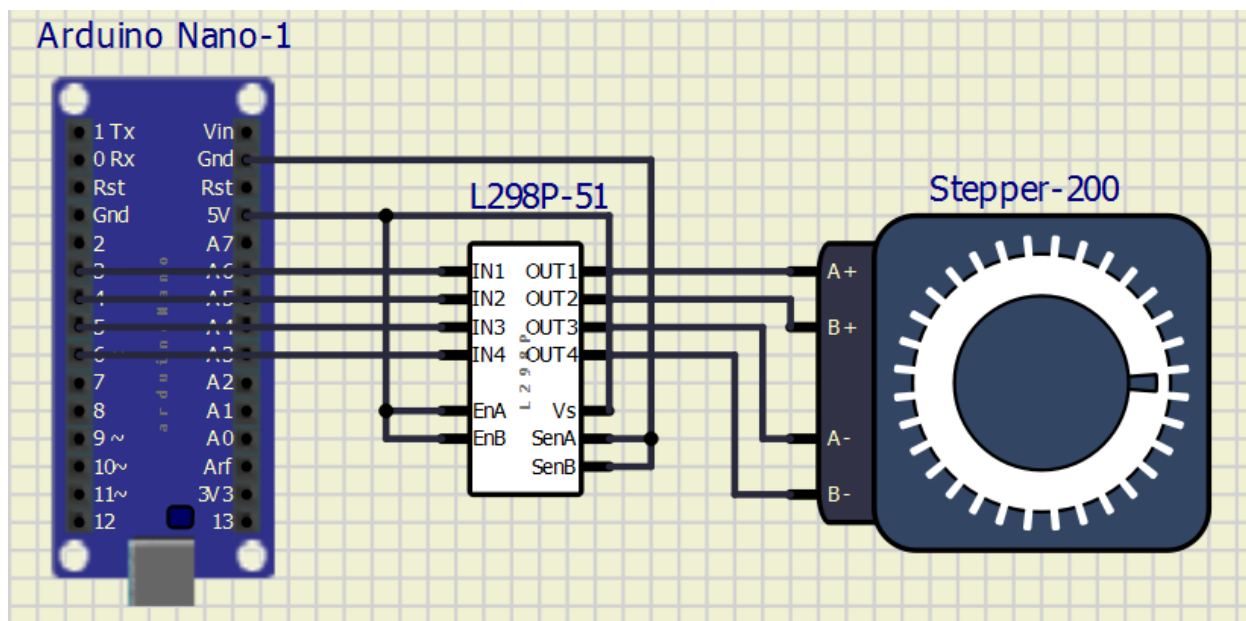


Рисунок 4.6. Подключение драйвера шагового двигателя

Входы драйвера IN1...IN4 принимают управляющие сигналы от микроконтроллера, входы EnA, EnB являются разрешающими и для упрощения могут замыкаться на питание, вход Vs – вход питания, SenA, SenB – входы, подтягивающиеся к общей земле. Выходы OUT1...OUT4 осуществляют выдачу сигналов непосредственно на шаговый двигатель.

Для работы с драйвером L298P используется библиотека Stepper, в которой имеется пример программы для управления двигателем.

Микроконтроллер ESP32

Общий вид ESP32 представлен на рисунке 1.1. Данный микроконтроллер обладает следующими характеристиками:

- процессор Xtensa LX6 (32бит);
- малопотребляющий сопроцессор ULP;
- встроенная память: основное ОЗУ (520 Кбайт), ОЗУ (16 Кбайт);
- возможность подключения внешней памяти (FLASH и SRAM);
- контроллер прямого доступа к памяти (DMA) для 13 периферийных устройств;
- встроенный генератор на 8МГц;
- встроенный генератор на 150 КГц с низким энергопотреблением;
- возможность подключения внешнего кварцевого резонатора на 2МГц...40МГц;
- четыре 64-битных таймера общего назначения с 16-битным делителем;
- три сторожевых таймера;

- 34 порта ввода-вывода;
- возможность подключения большей части периферии к любому порту ввода-вывода;
- Wi-Fi (802.11 b/g/n);
- Bluetooth (v4.2 BR/EDR, а также BLE);
- АЦП с разрядностью 12 бит (число каналов – до 18);
- 2 ЦАП с разрядностью 8 бит;
- 3x UART;
- 4x SPI;
- 2x I2C;
- 2x I2S;
- 1x ведущий (Host) контроллер SD/eMMC/SDIO;
- 1x ведомый (Slave) контроллер SDIO/SPI;
- 1x контроллер Ethernet MAC с поддержкой IEEE 1588;
- 1x контроллер CAN 2.0;
- порт для подключения ИК-датчиков;
- 6 каналов ШИМ для управления электродвигателями (при помощи H-моста);
- 16 каналов ШИМ для управления яркостью светодиодов (или для других целей);
- датчик Холла;
- аппаратные ускорители AES, SHA, RSA и ECC.

Микроконтроллер ESP8266

Общий вид ESP8266 представлен на рисунке 3.1. Данный микроконтроллер обладает следующими характеристиками:

- процессор Xtensa L106 (32бит);
- встроенная память: основное ОЗУ (96 Кбайт);
- возможность подключения внешней памяти (FLASH и SRAM);
- контроллер прямого доступа к памяти (DMA) для 13 периферийных устройств;
- встроенный генератор на 8МГц;
- возможность подключения внешнего кварцевого резонатора на 2МГц...40МГц;
- два 64-битных таймера общего назначения с 16-битным делителем;
- 11 портов ввода-вывода;
- Wi-Fi (802.11 b/g/n);

- АЦП с разрядностью 12 бит (число каналов – 11);
- 2 ЦАП с разрядностью 10 бит;
- 1x UART;
- 1x SPI;
- 1x I2C;
- 1x I2S.

Микроконтроллер ArduinoUno/Nano

Общий вид ArduinoUno представлен на рисунке 4.7. Данный микроконтроллер обладает следующими характеристиками:

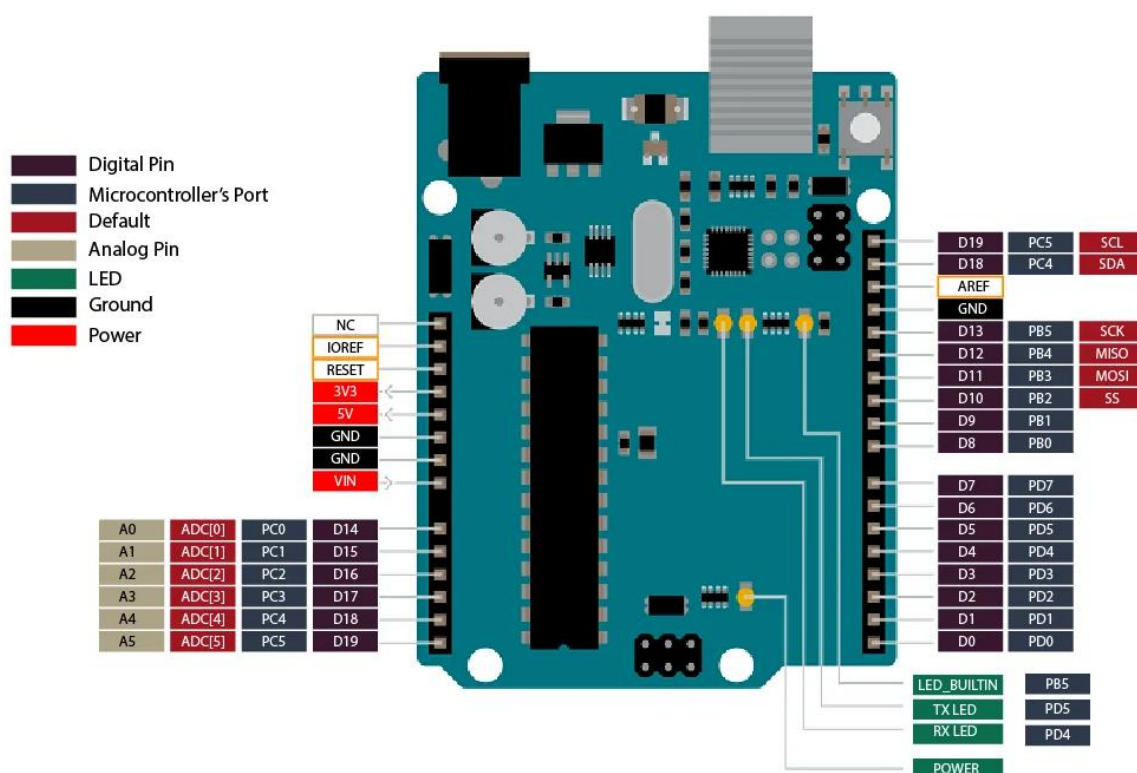


Рисунок 4.7. Распиновка микроконтроллера Arduino UNO

- Напряжение питания МК 3.. 5V (от 1.5V при пониженной частоте)
- Питание через стабилизатор 7.. 15V
- Макс. ток с пина 40 mA
- Мак. суммарный ток с пинов 200 mA
- Ток потребления От ~5 мкА (МК в режиме сна на модифицированной плате) до ~20 mA (на стоковой плате в обычном режиме)
- Частота процессора 16 MHz
- Flash память (программа) 32 кБ
- SRAM память (оперативная) 2 кБ

- EEPROM память 1 кБ
- Цифровые пины 20
- Аналоговые пины 8
- Аппаратные ШИМ пины 6

Символьный дисплей

В лабораторном макете используется символьный дисплей от компании «Амперка», предназначенный для работы с Тройка-модулями. Дисплей подключается к микроконтроллеру по интерфейсу I2C. Библиотека для работы с данным дисплеем называется «ST7032_asukiaaa.h».

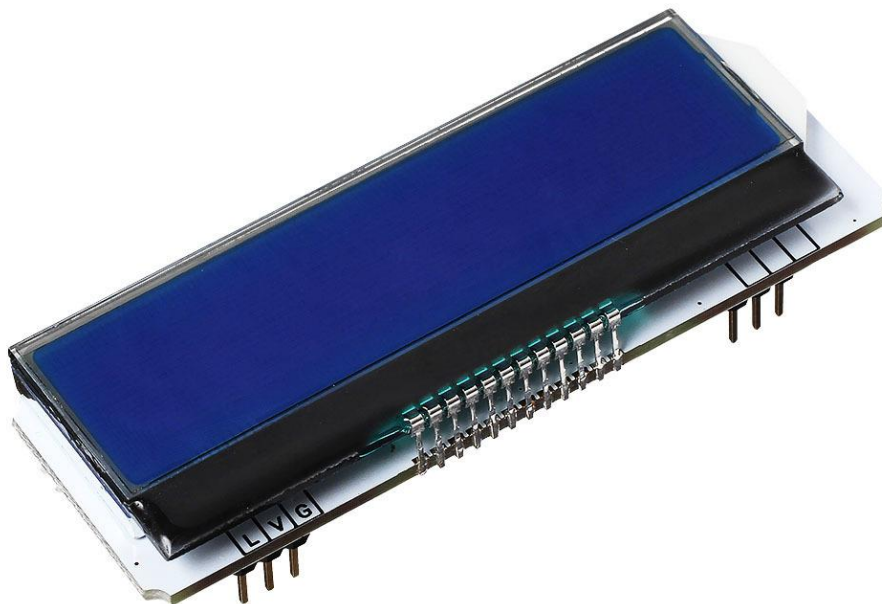


Рисунок 4.8. Символьный дисплей, используемый в стенде

Характеристики дисплея:

1. Тип дисплея: текстовый
2. Цвет: монохромный
3. Технология: LCD (Liquid Crystal Display)
4. Индикация: две строки по 16 символов
5. Драйвера матрицы: ST7032
6. Интерфейс: I2C
7. Адрес модуля: 0x3E
8. Тип подсветки: LED
9. Цвет подсветки: синий
10. Цвет символов: белый

11. Напряжение питания: 3.3–5 В

Основные команды для дисплея:

lcd.begin(16, 2); – указывается, сколько столбцов и строк в дисплее;

lcd.setContrast(50); – задание контраста фона и выводимых букв, может меняться от 0 до 63;

lcd.setCursor(0, 1); – ставит курсор печати на указанную позицию, в данном случае в нулевой столбец первой строки (нумерация начинается с нуля);

lcd.print(); – команда вывода информации на дисплей. Если вывести нужно число или значение переменной, то оно указывается в скобках, текст указывается в двойных кавычках.

Графический дисплей

В лабораторном стенде используется графический дисплей MOD-OLED-128x64 с разрешением 128x64 пикселя. Данный дисплей является монохромным и соединяется с микроконтроллером по шине I2C. Для работы с данным дисплеем используется библиотека «Adafruit_SSD1306.h».

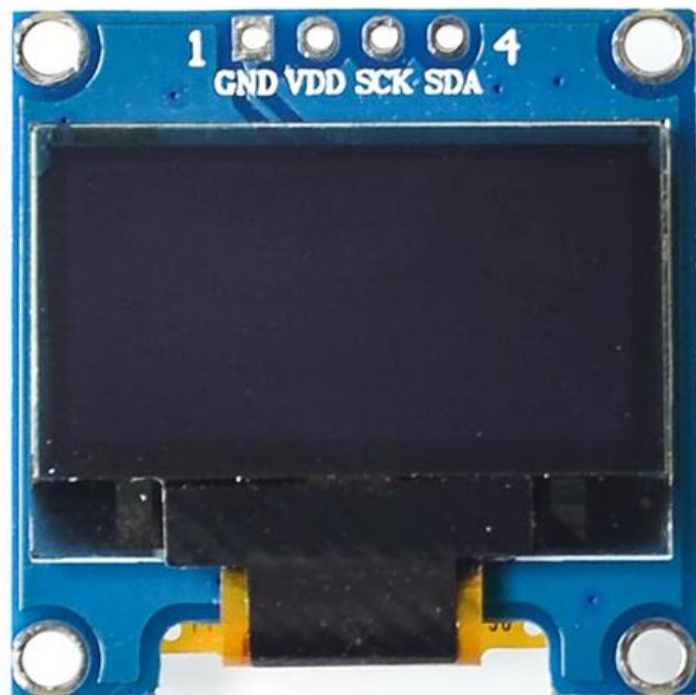


Рисунок 4.9. Графический дисплей, используемый в стенде

Характеристики дисплея:

1. Тип дисплея: графический
2. Цвет: монохромный
3. Технология: OLED
4. Разрешение: 128x64 пикселя
5. Драйвера матрицы: SSD1306
6. Интерфейс: I²C, SPI, UART (универсальный разъём UEXT)
7. Адрес модуля: 0x3C
8. Тип подсветки: LED
9. Цвета: чёрный/белый
10. Напряжение питания: 3.3V

Для графического дисплея существует довольно большой перечень команд. Ниже приведены основные команды, которые могут пригодиться при выполнении лабораторных и практических заданий.

display.setTextSize(размер текста); – определяет размер выводимого текста;

display.setTextColor(цвет); -определяет цвет текста (BLACK/WHITE);

display.setCursor(x,y); – ставит курсор печати в указанные координаты;

display.println(“ текст”); – выводит текст в кавычках на дисплей, если вывести необходимо числа или значения переменных, то они пишутся без кавычек;

display.display(); – команда на вывод информации на дисплей (все остальные команды носят предварительный характер);

display.clearDisplay(); – очистить буфер обмена с дисплеем (применяется перед сменой выводимой информации);

display.write(код символа); – команда отобразить символ в кодировке ASCII, в скобках указывается номер символа в кодировке);

display.drawRect(x, y, ширина, высота, цвет); – нарисовать прямоугольник (x и y – координаты верхнего левого угла);

display.fillRect(x, y, ширина, высота, цвет); – нарисовать закрашенный прямоугольник;

display.drawRoundRect (x, y, ширина, высота, радиус скругления, цвет); – нарисовать прямоугольник со скруглёнными углами;

display.fillRoundRect (x, y, ширина, высота, радиус скругления, цвет); – нарисовать закрашенный прямоугольник со скруглёнными углами;

display.drawCircle(x, y, радиус); – нарисовать окружность, x и y – координаты центра окружности;

display.fillCircle(x, y, радиус); – нарисовать закрашенную окружность;

display.drawTriangle(x1, y1, x2, y2, x3, y3, цвет); – нарисовать треугольник (1, 2, 3 – координаты вершин);

display.fillTriangle(x1, y1, x2, y2, x3, y3, цвет); – нарисовать закрашенный треугольник;

display.drawLine(x1, y1, x2, y2, цвет); -нарисовать линию;

display.drawPixel(x,y,цвет); – закрасить пиксель.

Таймеры

О таймерах рассказывается во второй лабораторной работе. Здесь указаны команды объявления таймера.

hw_timer_t *My_timer = NULL; -объявление таймера с именем My_timer. Команда даётся в самом начале программы до подпрограммы установки (setup) и основного цикла (loop).

void IRAM_ATTR onTimer(){ тело функции } – объявление функции с названием onTimer, которая будет вызываться в момент срабатывания таймера. IRAM_ATTR – параметр, который указывает на то, что функция будет использоваться с прерываниями.

Все дальнейшие команды пишутся в подпрограмме установки (setup).

My_timer = timerBegin(0, 80, true); – таймеру с именем My_timer выделяется 0 таймер микроконтроллера (всего их может быть 4 64-битных), предделитель этого таймера устанавливается 80 (то есть входную тактовую частоту 80 МГц предделитель будет уменьшать в 80 раз). Таким образом, на выходе предделителя частота тактовых импульсов составит 1 МГц. Третий параметр, который указан как true определяет тип счётчика: true – суммирующий счётчик, считающий от меньшего к большему, false – вычитающий счётчик.

timerAttachInterrupt(My_timer, &onTimer, true); – указывается, что таймер с именем My_timer при своём срабатывании генерирует прерывание, которое вызывает функцию-обработчик onTimer. Проще говоря, здесь указывается, что при срабатывании таймера My_timer вызывается функция onTimer.

timerAlarmWrite(My_timer, 5000000, true); – указывается, что счётчик таймера My_timer должен вести счёт до 5000000. При достижении указанного числа таймер работает и генерирует сигнал прерывания. Поскольку на счётчик поступают тактовые импульсы с предделителя, который задаёт их частоту в 1 МГц или 1000000 Гц (то есть за одну секунду на счётчик приходит один миллион импульсов), то получается, что, до 5000000 счётчик досчитает за 5 секунд. Последний параметр отвечает за сброс счётчика при достижении указанного значения: true – счётчик сбрасывается и дальше снова ведёт счёт (таймер будет срабатывать

каждые 5 секунд), false – счётчик один раз досчитав до указанного значения останавливается (таймер срабатывает однократно).

timerAlarmEnable(My_timer); – разрешение работы таймера My_timer.

Внешние прерывания

Внешние прерывания – это прерывания, событиями которых являются внешние сигналы, приходящие на пины микроконтроллера. Таких сигналов существует пять видов: низкий уровень сигнала (LOW), высокий уровень сигнала (HIGH), восходящий фронт (RISING), спадающий фронт (FALLING), изменение уровня (CHANGE).

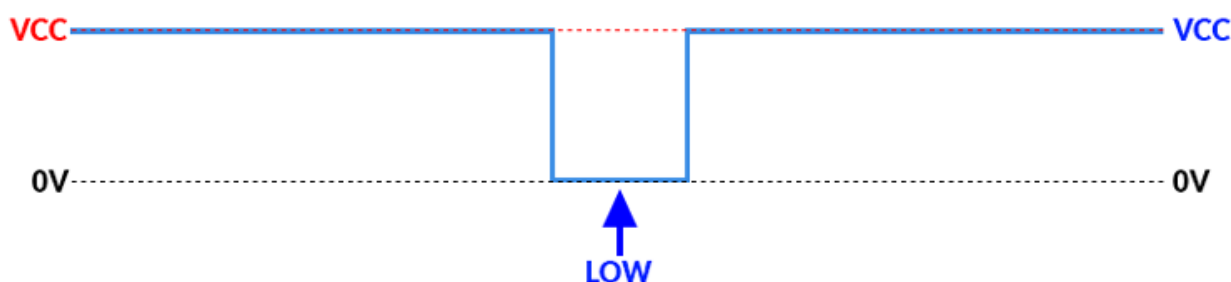


Рисунок 4.10. Сигнал прерывания по низкому уровню (LOW)

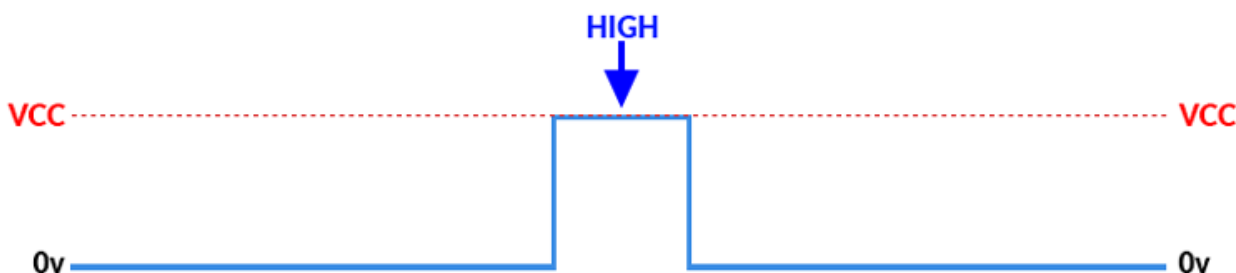


Рисунок 4.11. Сигнал прерывания по высокому уровню (HIGH)

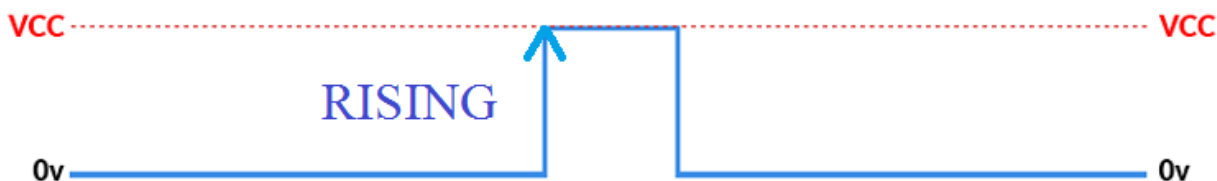


Рисунок 4.12. Сигнал прерывания по восходящему фронту (RISING)

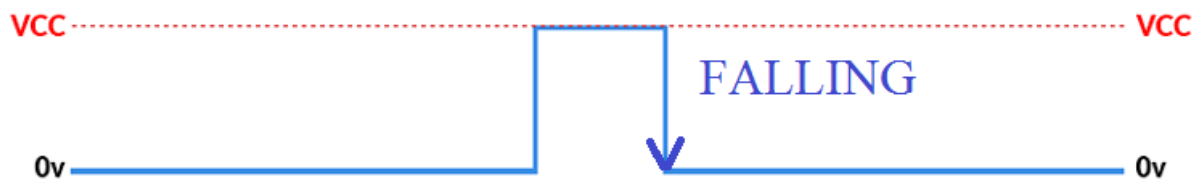


Рисунок 4.13. Сигнал прерывания по спадающему фронту (FALLING)

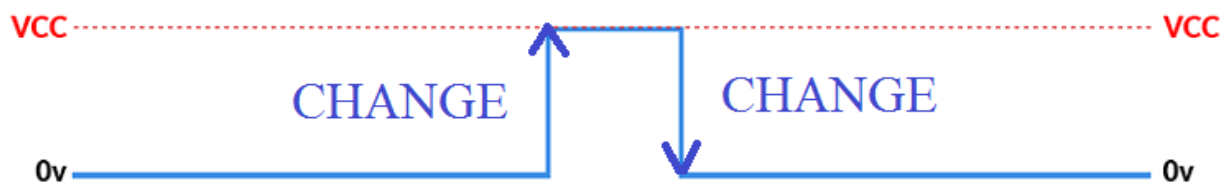


Рисунок 4.14. Сигнал прерывания по изменению уровня (CHANGE)

Для объявления внешних прерываний используются следующие команды:

void IRAM_ATTR function_name() {тело функции} – объявление функции `function_name`, которая вызывается при считывании сигнала прерывания. `IRAM_ATTR` – параметр, который указывает на то, что функция будет использоваться с прерываниями.

В подпрограмме установщика (setup) указывается команда:

attachInterrupt(номер пина, function_name, RISING); – объявление (добавление) прерывания, где в скобках на первой позиции указывается номер пина, с которого будет считываться внешний сигнал прерывания. На второй позиции указано название функции-обработчика прерывания, которая начнёт выполняться при появлении события прерывания. На последней позиции указывается тип сигнала, который является событием прерывания.

СПИСОК ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Богаченков А.Н. Цифровые устройства и микропроцессоры. Лабораторные работы 1, 2, 3 [Электронный ресурс]: Методические указания по выполнению лабораторных работ. – М.: МИРЭА, 2016 // https://www.mirea.ru/upload/sveden-new/11.05.01_RSK/metod_11.05.01_RSK_CUMP_ch2_2021.docx
2. Новожилов О.П. Основы микропроцессорной техники. В 2 т. – М.: ИП Радио-софт, 2007. Т. 1. – 432 с.
3. Блум Джереми. Изучаем Arduino: инструменты и методы технического волшебства: Пер. с англ. – СПб.: БХВ-Петербург, 2017. – 336 с.
4. Использование прерываний в модуле ESP32 [Электронный ресурс]: информационный ресурс «Мир микроконтроллеров» // <https://microkontroller.ru/esp32-projects/ispolzovanie-preryvanij-v-module-esp32/>