



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ № 1

**Общее знакомство с микроконтроллером ESP32
и средой разработки программного обеспечения**

Цифровые устройства и микропроцессоры

(наименование дисциплины (модуля) в соответствии с учебным планом)

Выполнил

Фамилия Имя Отчество

(ФИО)

Группа

XXXX-YY-ZZ

(шифр)

Преподаватель

Литвинов С.В.

(ФИО)

Институт

Радиоэлектроники и информатики

(краткое и полное наименование)

Кафедра

Радиоэлектронных систем и комплексов

(краткое и полное наименование кафедры, реализующей дисциплину (модуль))

Проверено

« »

20

г.

(подпись)

Москва 2022 г.

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

Код выполняется корректно, что демонстрирует график, изображенный на рисунке 1.

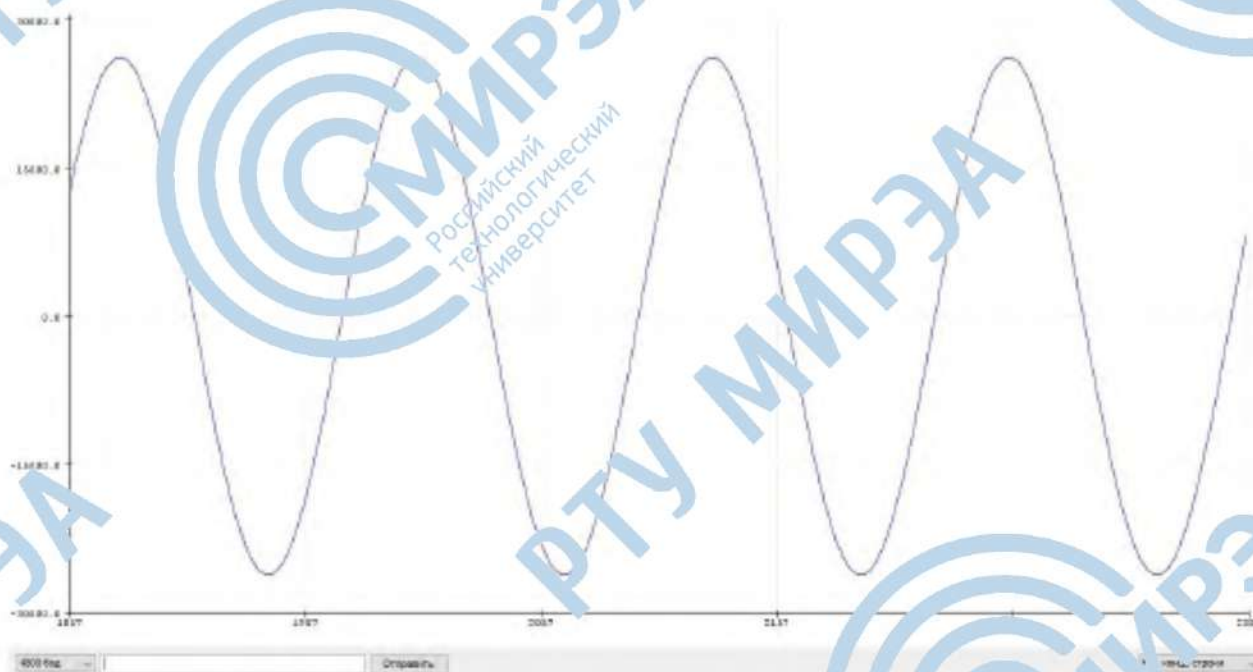


Рисунок 1. График на плоттере после выполнения программы LAB_1_1

Изменив значение переменной «А» так, чтобы оно стало больше 1, был получен новый график сигнала, зафиксированный на рисунке 2.



Рисунок 2. График на плоттере после выполнения программы LAB_1_1 с измененным значением переменной А

Используя функцию `ssat`, которая представлена на рисунке 3, удалось компенсировать ошибки при значении амплитуды больше 1, о чем свидетельствует график на рисунке 4.

```
LAB_1_1
1 int16_t res; //целочисленная шестнадцатиразрядная переменная, благодаря ей будем наблюдать эффект переполнения
2 float A=1.5; // задаем амплитуду сигнала
3 float shag=0.05; // шаг (разрешение)
4 float t0 = -3; // задаем начальное значение t=0
5
6 int ssat(float x, int over) // функция ssat, x - значение на входе, over - разрядность
7
8 void setup() {
9   Serial.begin(4800); //задаем скорость обмена данными с портом
10 }
11
12 void loop() {
13   for(int t=t0<10000;t++){
14     res=exp(t0-shag*t); //вычисляем значение нашей функции в цикле и мы таблица на весь числовой диапазон
15     Serial.print(res);
16   }
17   delay(1); //даем задержку, чтобы значения функции на графике выходили корректно
18 }
19
20
21 int ssat(float x, int over){
22   int K = 0;
23   int predel = pow(2,over)/2;
24   if(x>predel-1){
25     K = predel-1;
26   }
27   if(x < -1*predel){
28     K = -1*predel;
29   }
30   if ((x > -1*predel ) && (x < predel-1))
31   {
32     K = x;
33   }
34   return (K);
35 }
```

Рисунок 3. Использование функции `ssat` для компенсации ошибок



Рисунок 4. График на экране при запуске программы `LAB_1_1`

2. Построение графиков функций стандартной библиотеки C++. Вариант

2.3.

Измененный фрагмент программы представлен на рисунке 5. Соответствующий программе график с плоттера представлен на рисунке 6.

```
LAB1_1
1 int16_t test; //целочисленная шестнадцатизначная переменная, благодаря ей будем наблюдать эффект переполнения
2 float A=1.5; //шагем функции - амплитуда
3 float shag=0.01; //шагем шаг (период отсчета)
4 float t0 = 0; //даём начальное значение переменной
5
6 int test(float in, over); //функция, которая - значение на входе, over - разрядности
7
8 void setup() {
9   Serial.begin(4800); //задаём скорость обмена данными с портом
10 }
11
12 void loop() {
13   test(t0, over); //вызов функции
14   //выводим значение функции в цикл, который обходит весь числовой диапазон
15   //Serial.println(test);
16   t0 += shag;
17   //здесь можно добавить задержку, чтобы график выглядел корректно
18 }
```

Рисунок 5. Фрагмент кода для варианта 2.3.

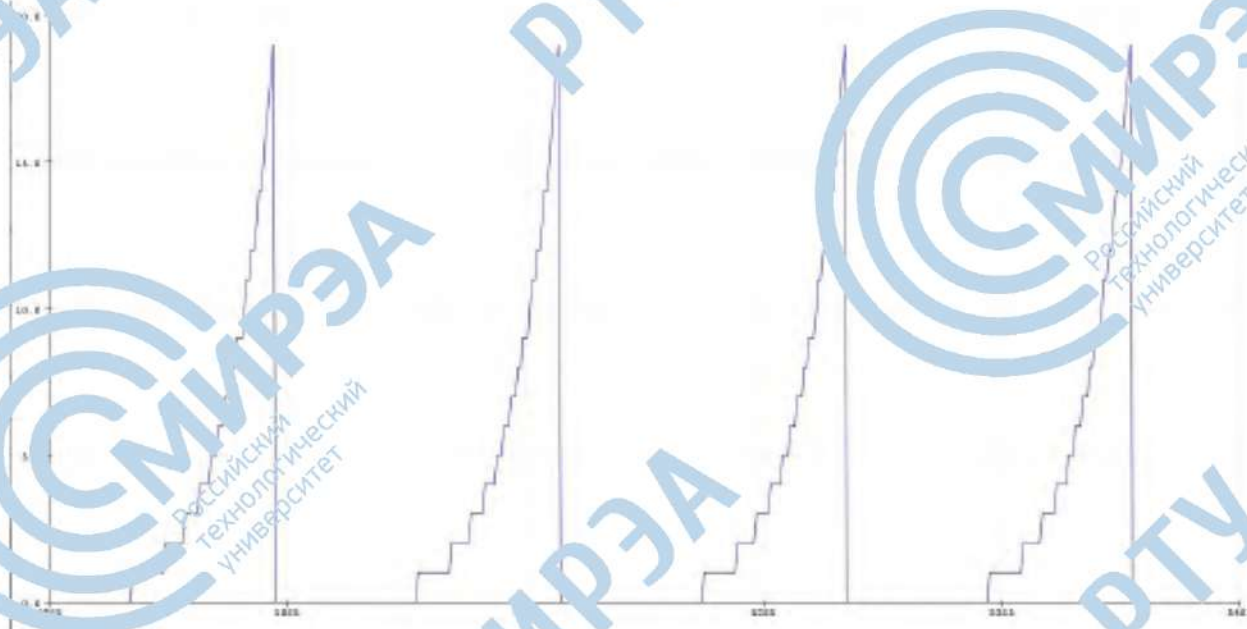


Рисунок 6. График для варианта 2.3.

3. Изучение цифровых входов/выходов. Вариант 3.

На рисунке 7 представлен доработанный код программы для варианта 3.

```
DIGITAL_OUT
1 void setup() {
2   pinMode(5, INPUT_PULLUP); // подключить 5 пин как вход, подтянутый к питанию
3   pinMode(19, INPUT_PULLUP);
4   pinMode(18, INPUT_PULLUP);
5   pinMode(13, OUTPUT); // подключить 13 пин как выход
6   pinMode(23, OUTPUT);
7 }
8
9
10 void loop() {
11   if (digitalRead(5) == LOW) // проверка состояния на 5 пине
12   {
13     digitalWrite(13,1); // выдать на 13 пин высокий уровень сигнала
14   }
15   else
16   {
17     digitalWrite(13,0); // выдать на 13 пин низкий уровень сигнала
18   }
19   if (digitalRead(19) == 0)
20   {
21     digitalWrite(23,1);
22   }
23   else
24   {
25     digitalWrite(23,0);
26   }
27
28   delay(100);
29
30   digitalWrite(13,0);
31   digitalWrite(23,0);
32   delay(100);
33   digitalWrite(13,1);
34   digitalWrite(23,1);
35   delay(100);
36
37   digitalWrite(13,0);
38   digitalWrite(23,0);
39   delay(100);
40   digitalWrite(13,1);
41   digitalWrite(23,1);
42   delay(100);
43
44 }
```

Рисунок 7. Код для варианта 3.

4. Изучение аналоговых входов/выходов

Модифицированная программа представлена на рисунке 8. Теоретические и практические значения кода представлены в таблице 1.

```
ANALOG_IN_OUT$
1 #include <ST7032_aculiana.h> //подключение библиотеки для работы с дисплеем
2 #ST7032_aculiana //обязательное подключение типа дисплея
3
4 int i; //переменная, где хранится значение, считанное АЦП
5
6
7 void setup() {
8   Serial.begin(115200); //начало работы с последовательным портом
9   pinMode(0, INPUT); //настройка пина 0 на вход
10   pinMode(1, INPUT); //настройка пина 1 на вход
11   digitalWrite(0, HIGH); //настройка пина 0 на высокий уровень
12   digitalWrite(1, HIGH); //настройка пина 1 на высокий уровень
13 }
14
15 float calcVelocity(int i)
16 {
17   return (i * 0.001) * 360.0 * 1.8;
18 }
19
20 void loop() {
21   int i = 0; //начало цикла
22   while (i < 10) {
23     //считывание значения сигнала АЦП, при входном коде 0
24     i = analogRead(0);
25     //вывод значения сигнала АЦП на 0 столбец 0 строки
26     Serial.println(i);
27     //считывание значения сигнала АЦП, при входном коде 1
28     i = analogRead(1);
29     //вывод значения сигнала АЦП на 1 столбец 0 строки
30     Serial.println(i);
31     //считывание значения сигнала АЦП, при входном коде 2
32     i = analogRead(2);
33     //вывод значения сигнала АЦП на 2 столбец 0 строки
34     Serial.println(i);
35     //считывание значения сигнала АЦП, при входном коде 3
36     i = analogRead(3);
37     //вывод значения сигнала АЦП на 3 столбец 0 строки
38     Serial.println(i);
39     //считывание значения сигнала АЦП, при входном коде 4
40     i = analogRead(4);
41     //вывод значения сигнала АЦП на 4 столбец 0 строки
42     Serial.println(i);
43     //считывание значения сигнала АЦП, при входном коде 5
44     i = analogRead(5);
45     //вывод значения сигнала АЦП на 5 столбец 0 строки
46     Serial.println(i);
47     //считывание значения сигнала АЦП, при входном коде 6
48     i = analogRead(6);
49     //вывод значения сигнала АЦП на 6 столбец 0 строки
50     Serial.println(i);
51     //считывание значения сигнала АЦП, при входном коде 7
52     i = analogRead(7);
53     //вывод значения сигнала АЦП на 7 столбец 0 строки
54     Serial.println(i);
55     //считывание значения сигнала АЦП, при входном коде 8
56     i = analogRead(8);
57     //вывод значения сигнала АЦП на 8 столбец 0 строки
58     Serial.println(i);
59     //считывание значения сигнала АЦП, при входном коде 9
60     i = analogRead(9);
61     //вывод значения сигнала АЦП на 9 столбец 0 строки
62     Serial.println(i);
63   }
64 }
```

Рисунок 8. Модифицированная версия программы

Таблица 1. Расчётные и измеренные показатели АЦП

Код АЦП	Код АЦП расчёт	U расчёт, В	Код АЦП измеренный	U изм, В
32	314	0,21	435	0,27
64	628	0,42	871	0,54
96	942	0,63	1307	0,81
128	1256	0,84	1743	1,08
160	1570	1,05	2179	1,36
192	1884	1,26	2615	1,64
224	2198	1,47	3051	1,92
255	4095	3,3	4095	3,3

В таблице 2 зафиксирован код с АЦП.

Таблица 2. Влияние шумов на работу АЦП

	Код АЦП
Измерение 1	1895
Измерение 2	1905
Измерение 3	1920
Измерение 4	1934
Измерение 5	1958

Расчёт шумовых разрядов:

$$1934 - 1895 = 39 = 100111_2, 6 \text{ разрядов}$$

5. Генерация широтно-импульсно модулированного сигнала

Текст модифицированной программы представлен на рисунке 9. Рисунок 10 демонстрирует график для ШИМ (синий) сигнала и сигнала с ЦАП (красный).

```

1 const int freq = 50; //задали частоту ШИМ-сигнала в Гц
2 const int ledChannel = 0; // выбор 0 канала ШИМ
3 const int resolution = 8; // разрешение ШИМ 8 – следовательно, период ШИМ сигнала можно разделить на 255
4 int led = ADC = 0;
5
6 void setup() {
7   Serial.begin(115200);
8   ledSetup(ledChannel, resolution); //брали в конфигурацию все параметры ШИМ сигнала
9   ledAttachPin(ledChannel); //присоединили к ШИМ 13 пин
10  pinMode(38, INPUT);
11  pinMode(13, OUTPUT);
12 }
13
14 // Функция для генерации ШИМ сигнала
15 void ledSetup(int channel, int resolution) {
16   pinMode(channel, OUTPUT);
17   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
18   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
19   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
20   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
21   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
22   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
23   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
24   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
25   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
26   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
27   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
28   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
29   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
30   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
31   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
32   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
33   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
34   // Функция для генерации ШИМ, dutyCycle – это коэффициент заполнения
35 }
36 }

```

Рисунок 9. Текст модифицированной программы

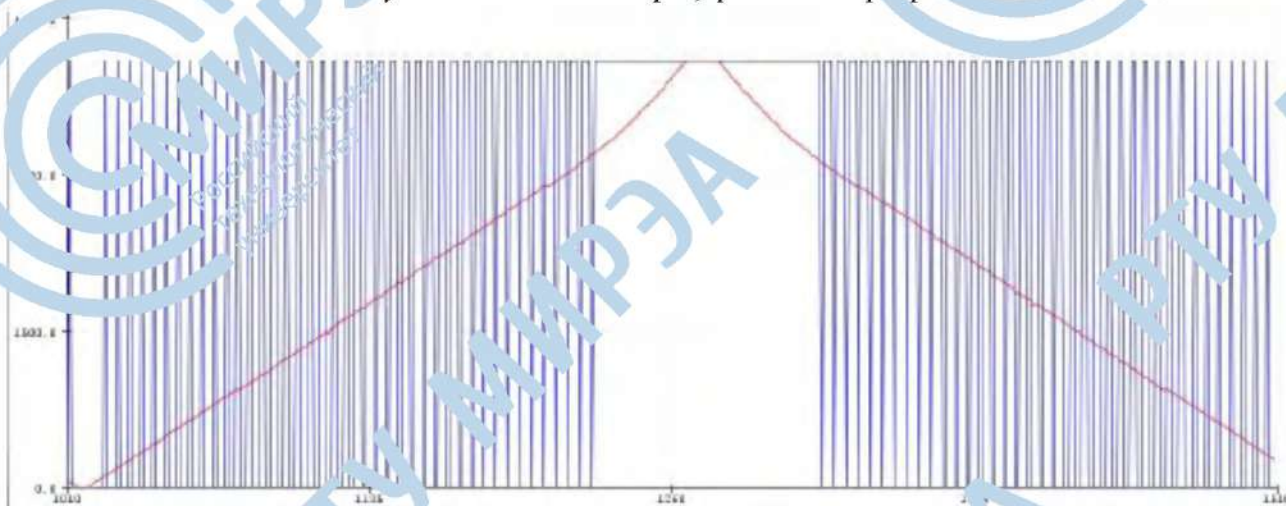


Рисунок 10. График для ШИМ (синий) сигнала и сигнала с ЦАП (красный)