МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Кафедра лазерных и биотехнических систем

Пояснительная записка

к курсовой работе

«Устройство считывания данных для непрерывного мониторинга уровня лактата»

По направлению подготовки 12.03.04
Биотехнические системы и технологии
(уровень бакалавриата)

Выполнил студент группы 6364–120304D_____Д. В. Сокольский

Руководитель проекта _____Д. В. Корнилин

Работа защищена с оценкой _____

Самара 2023

ЗАДАНИЕ

Разработать устройство считывания данных для непрерывного мониторинга уровня лактата

- Тип датчика — амперометрический (определение вольт-амперной характеристики)
- Диапазон регистрируемых токов от 1 нА до 1 мкА
- Диапазон задаваемых напряжений от 0 до 300 мВ
- Погрешность задания напряжения и установки значения тока 1%
- Предусмотреть термокомпенсацию результатов измерений
- Интервал между измерениями — 1 мин
- Сохранение результатов за последние 24 ч во встроенной памяти
- Питание — часовая батарейка SR626SW
- Время автономной работы не менее 14 дней
- Передача данных по интерфейсу Bluetooth

РЕФЕРАТ

Пояснительная записка: 32 страницы, 14 рисунков, 7 источников, 3 приложения.

ЛАКТАТ, УСТРОЙТСТВО СЧИТЫВАНИЯ ДАННЫХ, МИКРОКОНТРОЛЛЕР, BLUETOOTH, STM32WB55RCV6

В курсовом проекте разработаны структурная и принципиальная схемы устройства считывания данных для непрерывного мониторинга уровня лактата, осуществлен выбор микроконтроллера с интегрированным блоком Bluetooth и АЦП. Разработан алгоритм анализа данных и реализующая его программа на языке Си.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Лактат — это продукт клеточного метаболизма, соль молочной кислоты, образующаяся при замещении иона водорода молочной кислоты на Na+ или К+. Измерение концентрации лактата является общепринятым тестом во всем мире. В первую очередь быстрый анализ лактата необходим в отделениях реанимации, скорой помощи, хирургии, травматологии, роддомах и перинатальных центрах, а также в эндокринологических отделениях и спортивной медицине. Доказана роль оценки уровня лактата крови у больных, находящихся в критическом состоянии, в качестве:

1. Показателя кислородной задолженности тканей;

2. Показателя эффективности проводимой терапии;

3. Прогностического признака неблагоприятного исхода.

Концентрация лактата — это показатель кислородной задолженности тканей при целом ряде состояний. Выделяют следующие причины гиперлактемии:

1. Наиболее вероятные: сепсис, кардиогенный шок, полиорганная недостаточность, кислородное голодание.

2. Значимые:большие судорожные припадки, инфаркт кишечника, введение адреналина или натрия нитропруссида.

3. Требующие внимания:дефицит витамина В1(тиамина), алкалоз.

4. Маловероятные:анемия, заболевания печени. [1]

В данном курсовом проекте рассматривается способ создания устройства на базе микроконтроллера, который сможет регистрировать уровень лактата в межклеточном веществе. В процессе был подобран микроконтроллер с интегрированным модулем Bluetooth, необходимые усилители, карта памяти, а также написана управляющая программа на языке Си.

1 РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ УСТРОЙСТВА

Структурная схема устройства представлена на рисунке 1.

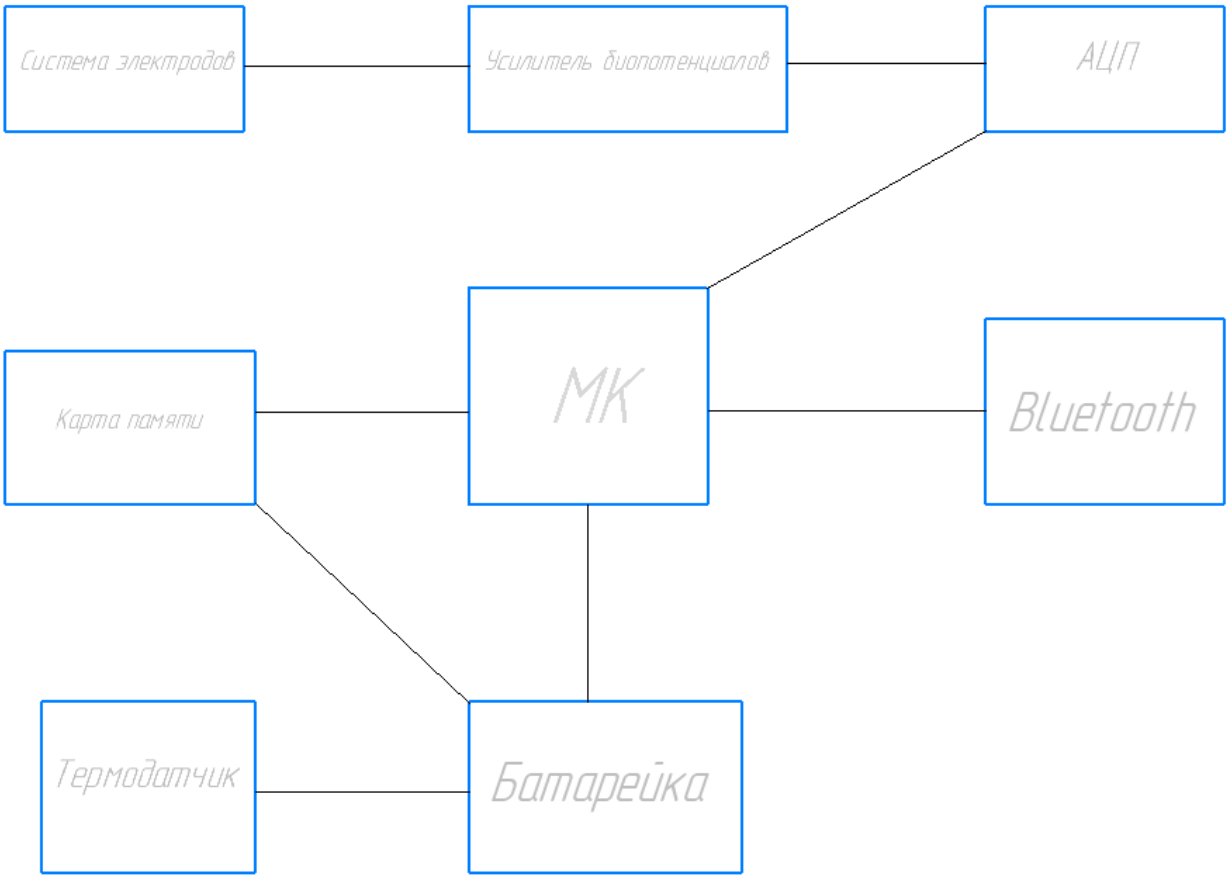


Рисунок 1 – Структурная схема устройства

Принцип работы устройства заключается в следующем: через электроды сигнал передается на усилитель биопотенциалов, который усиливает амплитуду сигнала. Эти данные поступают в микроконтроллер, где проходят первичную обработку, и с помощью алгоритма на языке Си анализируются. Помимо анализа, данные передаются по модулю Bluetooth, интегрированному в микроконтроллер, и записываются в карту памяти.

2 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ СХЕМЫ УСТРОЙСТВА

Электрическая принципиальная схема представлена в приложении.

2.1 Выбор усилителя биопотенциалов

Схема подает напряжение, чтобы вызвать окислительно-восстановительную реакцию на рабочем электроде. Этот заряд усиливает токовую реакцию электрода, и, наконец, чувствительная схема усиливает сигнал электрода. Противоэлектрод (CE) находится под отрицательным напряжением относительно рабочего электрода. Кроме того, ток, генерируемый на выводе рабочего электрода, составляет менее 100 нА при концентрации мг/дЛ. Поскольку этого тока недостаточно, для преобразования тока в выходное напряжение требуется усилитель трансимпеданса (TIA) с чрезвычайно низким током смещения. С этой целью мы использовали операционный усилитель MAX9913, который подходит для аналогичных применений. При комнатной температуре имеет низкий ток смещения и хорошие характеристики защиты от шумовых помех. Типичная схема работы инструментального усилителя из даташитаMAX9913 приведена на рисунке 2.[7]

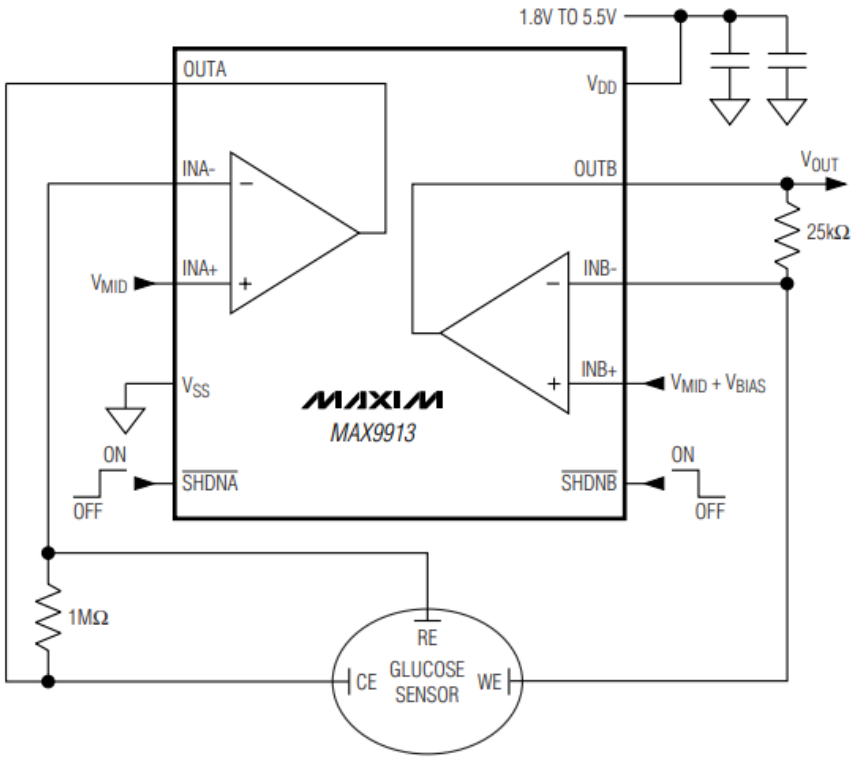


Рисунок 2 – Типичная схема работы инструментального усилителя

Нумерация и назначение выводов AD623 приведено ниже (рисунки 3, 4).

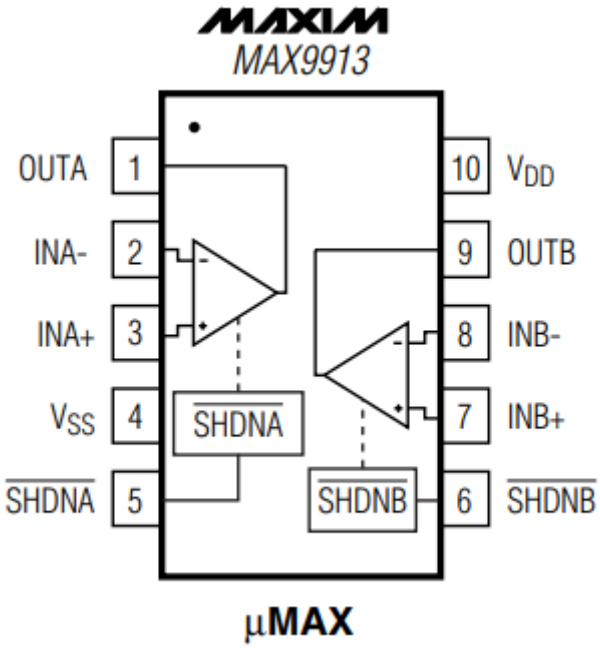


Рисунок 3 – Нумерация выводов

PIN				NAME	FUNCTION
MAX9910	MAX9911	MAX9912	MAX9913		
1	1	—	—	IN+	Noninverting Amplifier Input
2	2	4	4	V_{SS}	Negative Supply Voltage
3	3	—	—	IN-	Inverting Amplifier Input
4	4	—	—	OUT	Amplifier Output
5	6	8	10	V_{DD}	Positive Supply Voltage
—	5	—	—	$\overline{SHDN}$	Shutdown
—	—	1	1	OUTA	Amplifier Output Channel A
—	—	2	2	INA-	Inverting Amplifier Input Channel A
—	—	3	3	INA+	Noninverting Amplifier Input Channel A
—	—	—	5	$\overline{SHDNA}$	Shutdown Channel A
—	—	—	6	$\overline{SHDNB}$	Shutdown Channel B
—	—	5	7	INB+	Noninverting Amplifier Input Channel B
—	—	6	8	INB-	Inverting Amplifier Input Channel B
—	—	7	9	OUTB	Amplifier Output Channel B

Рисунок 4 – Назначение выводов

2.2Выбор микроконтроллера

С учетом технического задания микроконтроллер должен обладать следующими свойствами:

- Встроенный аналого-цифровой преобразователь;

- Для передачи данных по Bluetooth: встроенный стек протокола Bluetooth;

- Свободные выводы для подключения карты памяти.

Для решения задачи был выбран микроконтроллер STM32WB55RCV6 фирмы ST Microelectronics [4].STM32WB55 содержит два производительных ядра ARM-Cortex:

- ядро ARM® -Cortex® M4 (прикладное), работающее на частотах до 64 МГц, для пользовательских задач имеется модуль управления памятью, модуль плавающей точки, инструкции ЦОС (цифровой обработки сигналов), графический ускоритель (ART accelerator);
- ядро ARM®-Cortex® M0+ (радиоконтроллер) с тактовой частотой 32 МГц, реализующее низкоуровневые функции сетевых протоколов.

Данный микроконтроллер включает в себя все необходимые периферийные устройства, например,радиомодуль с поддержкой Bluetooth, диапазон питающего напряжения от 1,71 до 3,6 В.

Основные характеристики:

- типовое энергопотребление <53 мкА/МГц (при включюченныхRF и SMPS);

- потребление в режиме останова 2,1 мкА (радиочасть в режиме ожидания (standby));

- потребление в выключенном состоянии (Shutdown) 13 нА;

- диапазон допустимых напряжений питания 1,71…3,6 В (встроенный DC- DC–преобразователь и LDO-стабилизатор);

- рабочий температурный диапазон -40…105°C.

Структурная схема микроконтроллера приведена на рисунке 5, а назначение выводов портов корпуса на рисунке 6:

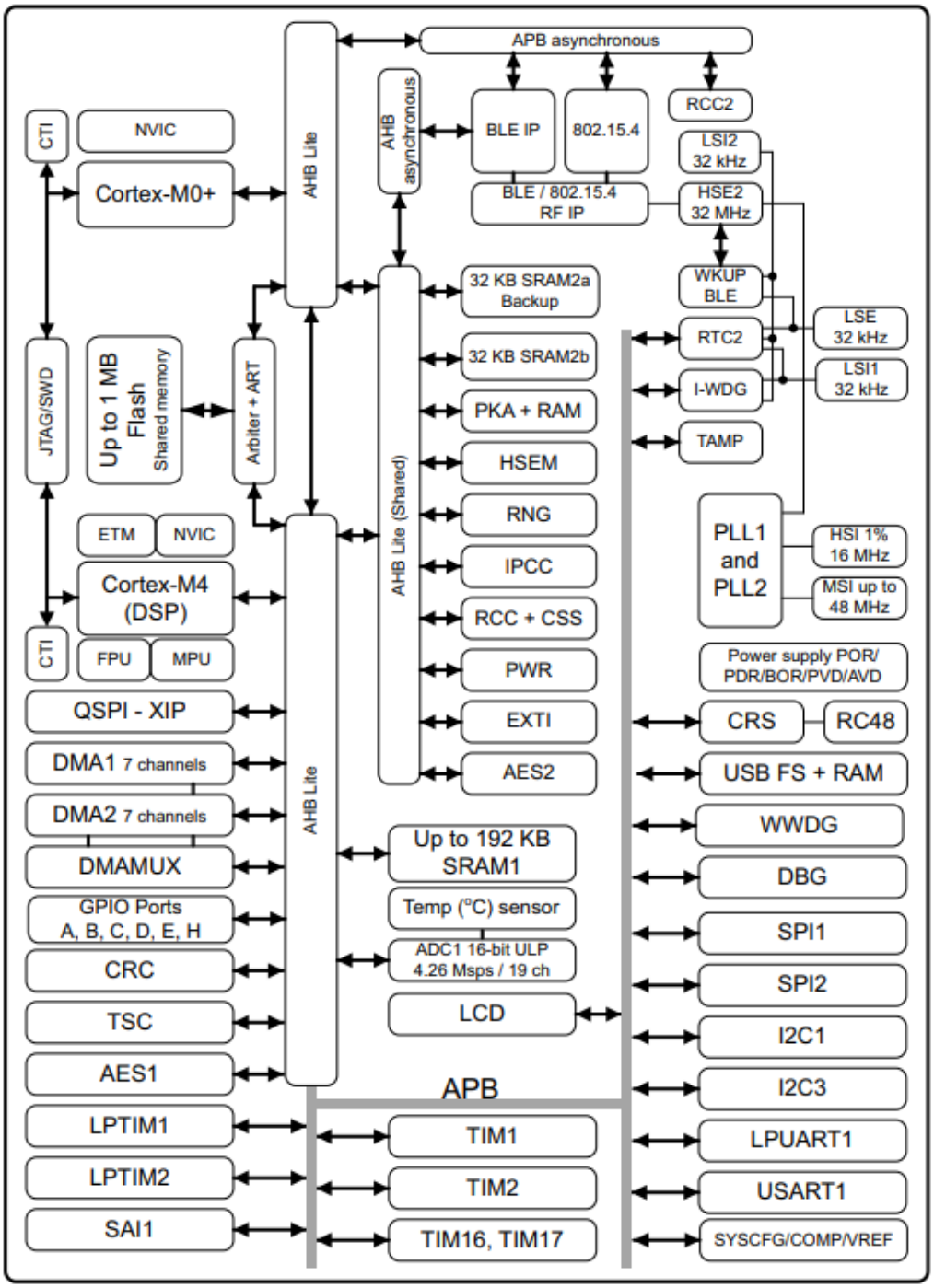


Рисунок 5 – Структурная схема

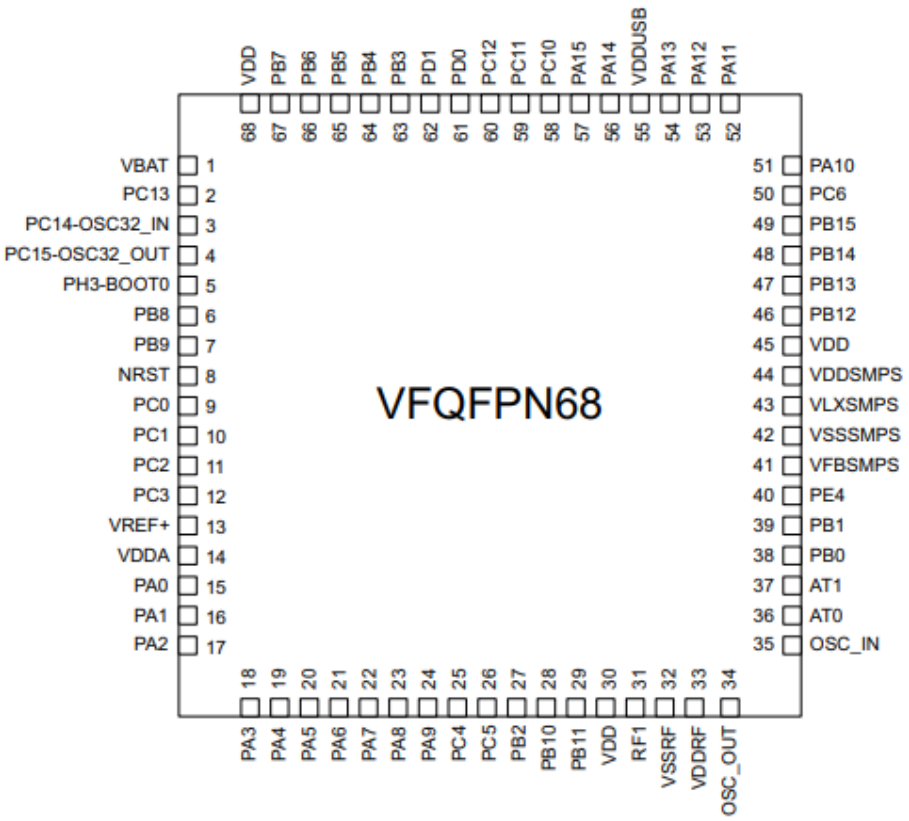


Рисунок 6 – Назначение выводов

Подключение будет осуществляться согласно типовой схеме из Applicationnote (рисунок 7):[5]

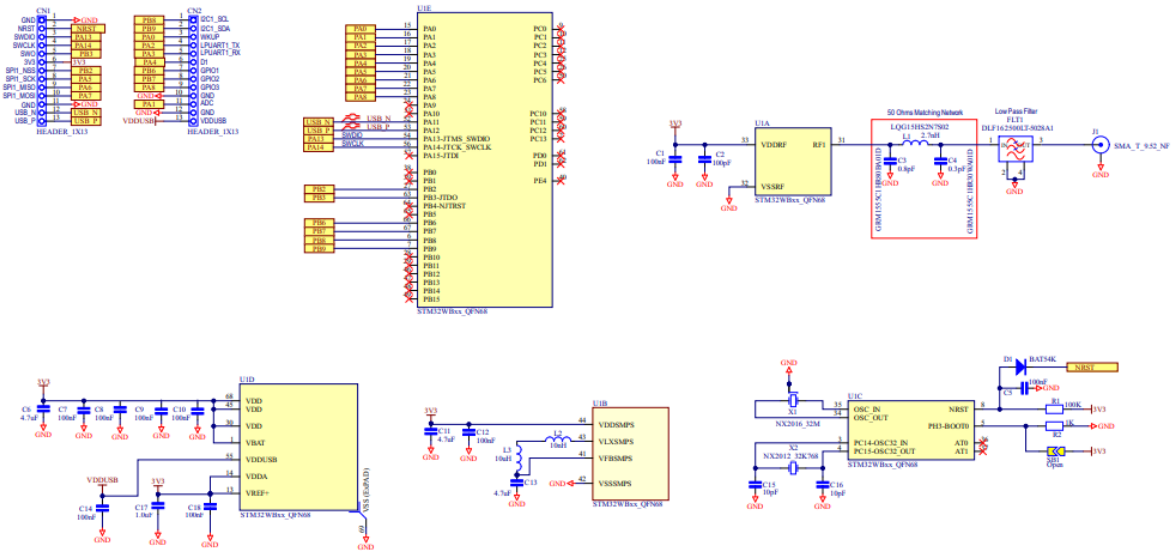


Рисунок 7 – Типовая схема подключения STM32WB55

2.3 Выбор термодатчика

Для точного измерения температуры окружающей среды (в том числе места контакта устройства с телом исследуемого) возникла необходимость

установки периферийного высокоточного датчика, который обеспечивал бы схему информацией о температуре и в случае перегрева предпринимались бы необходимые меры для исключения снятия неточных данных. Таковым был выбран датчик от фирмы TexasInstrumentsTMP102[2]. Его упрощенная схема и конфигурация выводов изображены на рисунках 8 и 9. Информацию с него будем получать по шине I^2C.

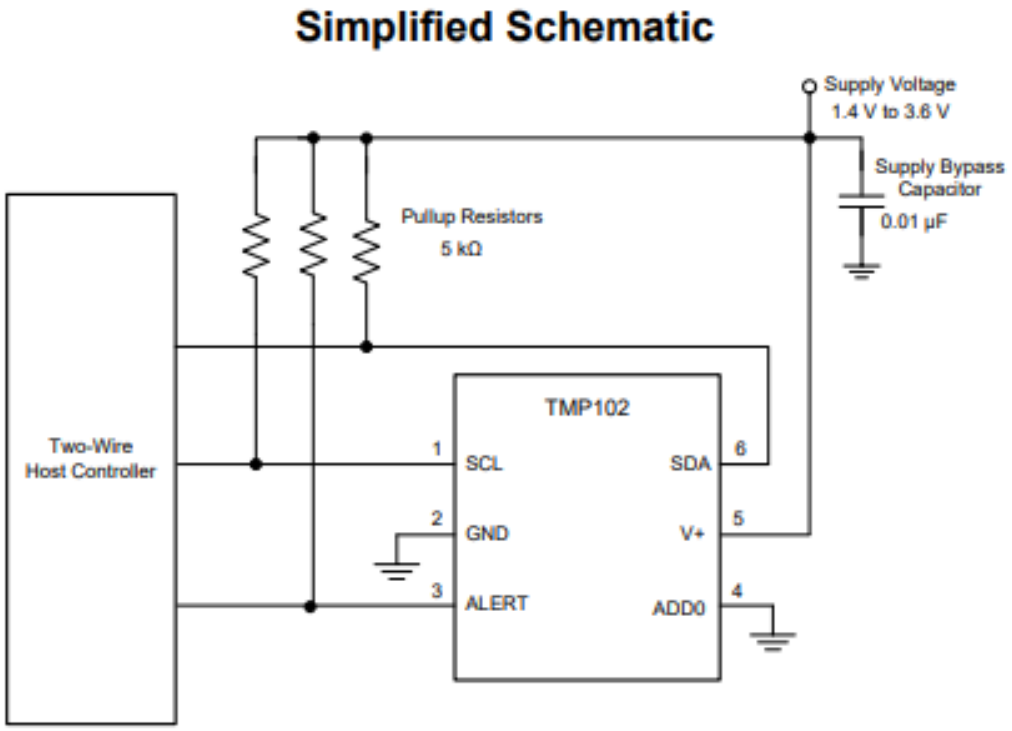
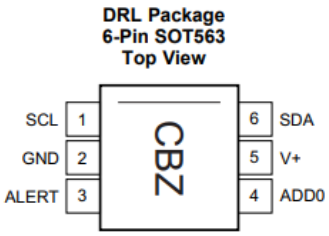


Рисунок 8 – Упрощенная схема TMP102



Pin Functions

PIN		I/O	DESCRIPTION
NO.	NAME		
1	SCL	I	Serial clock. Open-drain output; requires a pullup resistor.
2	GND	—	Ground
3	ALERT	O	Overtemperature alert. Open-drain output; requires a pullup resistor.
4	ADD0	I	Address select. Connect to GND or V+
5	V+	I	Supply voltage, 1.4 V to 3.6 V
6	SDA	I/O	Serial data. Open-drain output; requires a pullup resistor.

Рисунок 9 - Конфигурация выводовTMP102

2.4Схема питания

Питание данного устройства спроектировано от часовой батарейки с напряжением питания 1.8В. В связи с тем, что практически все элементы схемы питаются напряжением больше 3 вольт, используем DC-DC преобразователь напряжения. Возьмем модель от TexasInstruments TPS63802 2-A[2]. Конфигурация выводов изображена на рисунке 9. Данный конвертер поднимет напряжение с 1.8 вольт до 3.5.

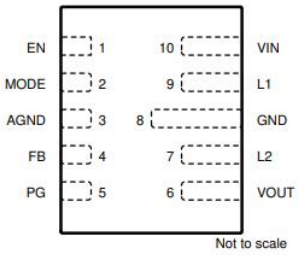


Figure 7-1. 10-Pin DLA Package (Top View)

Table 7-1. Pin Functions

PIN		DESCRIPTION
NAME	NO.	
EN	1	Device Enable input. Set HIGH to enable and LOW to disable. It must not be left floating.
MODE	2	PFM/PWM mode selection. Set LOW for power save mode, set HIGH for forced PWM mode. It must not be left floating.
AGND	3	Analog ground
FB	4	Voltage feedback sensing pin
PG	5	Power good indicator, open-drain output
VOUT	6	Power stage output
L2	7	Connection for inductor
GND	8	Power ground
L1	9	Connection for inductor
VIN	10	Supply voltage input

Рисунок 9 -Конфигурация выводов

В целях обеспечения сохранности первоначального вида сигнала, поставим LOW-DROPOUT регулятор напряжения перед питанием усилителя биопотенциалов. Возьмем R1517S331D-E2-FE от компании RICOH[3].На рисунке 10 показано типичное включение данного регулятора в цепь. Данный регулятор обезопасит усилитель от дребезгов питания.

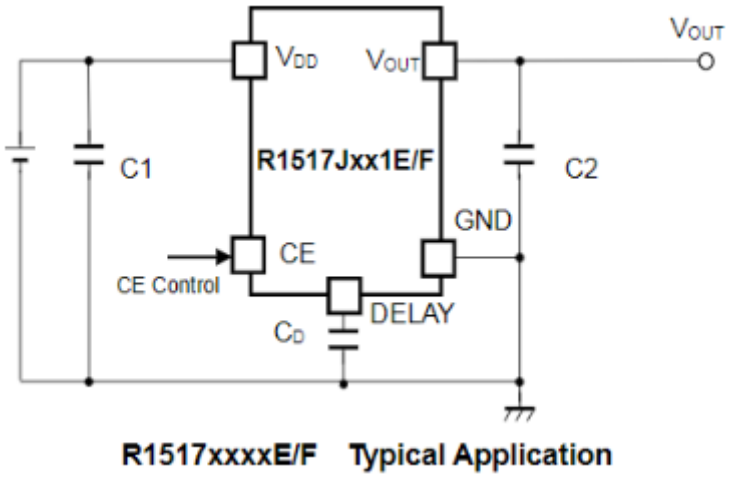


Рисунок 10 - Типичное включение R1517S331D-E2-FE

3 РАЗРАБОТКА ПРОГРАММЫ

3.1 Разработка алгоритма программы

Алгоритм начинается с включения питания и инициализации всех периферийных устройств и модулей. В инициализацию включено: Модуль SystemClock(задание тактовой частоты для всех остальных модулей), модуль TIM1(модуль, связанный с таймером, который задает скорость передачи данных), модуль ADC1 (АЦП),модуль TIM16 и TIM17, таймеры, которые будут отвечать за генерацию импульсов ШИМ,модуль GPIO (данный модуль отвечает за инициализацию дополнительных выводов микроконтроллера). После этого начинается основная программа. АЦП принимает значение сигнала, которые записываются в массив. Массив данных передается по Bluetoothи записывается на карту памяти. Основная программа возвращается на начало своего цикла.

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте рассмотрены принципы разработки устройства на базе микроконтроллера. Было разработано устройство считывания данных для непрерывного мониторинга лактата. Данные передаются по интерфейсу Bluetooth. Разработаны структурная и принципиальная схемы устройства, осуществлен выбор микроконтроллера и компонентов вспомогательных элементов схемы. Разработан алгоритм программы на языке Си.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Первушин Ю.В., Зинина А.Н., Рогова С.Ш. Лактат. Актуальность исследования и возможности определения. Краткое пособие для врачейлаборантов. Москва-Ставрополь, 2010.

2 TI. TMP102 [Электронныйресурс].URL:https://static.chipdip.ru/lib/747/DOC011747810.pdf(Датаобращения: 01.05.2023)

3 RICOH. R1517x CMOS-based LDO [Электронныйресурс].URL: https://pdf1.alldatasheet.com/datasheet-pdf/view/899043/RICOH/R1517S331D-E2-FE.html (Датаобращения: 01.05.2023)

4 ST. STM32WB55RCV6[Электронныйресурс].URL:https://www.st.com/resource/en/datasheet/stm32wb55cc.pdf (Датаобращения: 08.05.2023)

5 ST. AN5165Application note [Электронныйресурс].URL:How to develop RF hardware using STM32WB microcontrollers - Application note(Датаобращения: 21.05.2023)

6Texas Instruments. TPS63802 2-A, High-efficient, Low IQ Buck-boost Converter in DFN Package [Электронныйресурс].URL: https://static.chipdip.ru/lib/529/DOC010529777.pdf(Датаобращения: 28.05.2023)

7 Maxim Integrated Products. MAX9913, 200kHz, 4µA, Rail-to-Rail I/O Op Amps with Shutdown [Электронныйресурс].URL: https://doc.softelectronics.ru/docs/op3/MAX9913.pdf (Датаобращения: 28.05.2023)

ПРИЛОЖЕНИЕ А

```
/* USER CODE BEGIN Header */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "string.h"
#include "stdio.h"
#include "w25qxx.h"
#include "tmp102.h"
/* USER CODE END Includes */
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
/* USER CODE END PTD */
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */
/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */
/* USER CODE END PM */
/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
I2C_HandleTypeDef hi2c1;
IPCC_HandleTypeDefhipcc;
RTC_HandleTypeDefhrtc;
SPI_HandleTypeDef hspi1;
TIM_HandleTypeDef htim1;
TIM_HandleTypeDef htim16;
TIM_HandleTypeDef htim17;
UART_HandleTypeDef huart1;
/* USER CODE BEGIN PV */
tmp102_device dev;
char trans_str[64] = {0,};
volatile uint16_t adc = 0;
uint8_t databuffer[256]; // 256 байт = 1 страницапамяти
uint32_t buffer[3]; //массив для одного измерения
uint8_t send_ble[2]={0,0};
bool data_flg = false;
/* USER CODE END PV */
/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void PeriphCommonClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_ADC1_Init(void);
```

```
static void MX_I2C1_Init(void);
static void MX_IPCC_Init(void);
static void MX_RF_Init(void);
static void MX_RTC_Init(void);
static void MX_TIM1_Init(void);
static void MX_TIM16_Init(void);
static void MX_TIM17_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_SPI1_Init(void);
/* USER CODE BEGIN PFP */
/* USER CODE END PFP */
/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
   if(hadc->Instance == ADC1) //check if the interrupt comes from ACD1
   {
adc = HAL_ADC_GetValue(&hadc1);
HAL_SPI_Transmit(&hspi1, (uint8_t*)trans_str, strlen(trans_str), 1000);
adc = 0;
   }
}
void HAL_RTC_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
      if(htim->Instance == TIM1) //check if the interrupt comes from TIM1
      {
HAL_ResumeTick();
      }
/* USER CODE END 0 */
  * @brief  The application entry point.
  * @retval int
int main(void)
  /* USER CODE BEGIN 1 */
  /* USER CODE END 1 */
  /* MCU Configuration--------------------------------------------------------*/
  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
HAL_Init();
  /* Config code for STM32_WPAN (HSE Tuning must be done before system clock configuration) */
MX_APPE_Config();
  /* USER CODE BEGIN Init */
HAL_StatusTypeDef tmp102Initialize(tmp102_device* dev, I2C_HandleTypeDef *hi2c, TMP102ADDR i2c_addr);
  /* USER CODE END Init */
  /* Configure the system clock */
SystemClock_Config();
/* Configure the peripherals common clocks */
PeriphCommonClock_Config();
  /* IPCC initialisation */
MX_IPCC_Init();
```

```
  /* USER CODE BEGIN SysInit */
  /* USER CODE END SysInit */
  /* Initialize all configured peripherals */
MX_GPIO_Init();
  MX_ADC1_Init();
  MX_I2C1_Init();
MX_RF_Init();
MX_RTC_Init();
  MX_TIM1_Init();
  MX_TIM16_Init();
  MX_TIM17_Init();
  MX_USART1_UART_Init();
  MX_SPI1_Init();
  /* USER CODE BEGIN 2 */
  W25qxx_Init();
HAL_ADCEx_Calibration_Start(&hadc1, adc);
HAL_ADC_Start_IT(&hadc1);
HAL_TIM_Base_Start_IT(&htim1);
  ////////////////////////////////////////////////////////
HAL_UART_Transmit(DEBUG_UART, (uint8_t*)"Start W25QXX\n", 13, 1000);
    //W25qxx_EraseChip();
    W25qxx_EraseBlock(0); // 65536 байт
    //W25qxx_EraseSector(0); // 4096 байт
    uint8_t b0 = 's';
    uint8_t b1 = 't';
    uint8_t b2 = 'D';
    W25qxx_WriteByte(b0, 25);
    W25qxx_WriteByte(b1, 26);
    W25qxx_WriteByte(b2, 27);
    ////////////////////////////////////// ЧТЕН?ЕПОБАЙТНО ////////////////////////////////
    uint8_t buf[64] = {0,};
    W25qxx_ReadByte(&buf[0], 25);
    W25qxx_ReadByte(&buf[1], 26);
    W25qxx_ReadByte(&buf[2], 27);
HAL_UART_Transmit(DEBUG_UART, (uint8_t*)buf, strlen((char*)buf), 100);
HAL_UART_Transmit(DEBUG_UART, (uint8_t*)"\r\n", 2, 100);
    ////////////////////////////////////// ЗАП?СЬСТРАН?ЦЫ ////////////////////////////////
    uint8_t w_buf[] = "istarik.ru";
    W25qxx_WritePage(w_buf, 0, 28, 10);
    ////////////////////////////////////// ЧТЕН?ЕСТРАН?ЦЫ ////////////////////////////////
memset(buf, 0, 64);
    W25qxx_ReadPage(buf, 0, 28, 10);
    ////////////////////////////////////// ЗАП?СЬСЕКТОРА ////////////////////////////////
    W25qxx_WriteSector(w_buf, 0, 1350, 10);
////////////////////////////////////// ЧТЕН?Е СЕКТОРА ////////////////////////////////
W25qxx_ReadSector(buf, 0, 1350, 10);
```

```
  ////////////////////////////////////// ЗАП?СЬБЛОКА ////////////////////////////////
  W25qxx_WriteBlock(w_buf, 0, 9350, 10);
  ////////////////////////////////////// ЧТЕН?ЕБЛОКА ////////////////////////////////
  W25qxx_ReadBlock(buf, 0, 9350, 10);
////////////////////////////////////// ЧТЕН?Е ЛЮБОГО КОЛ?ЧЕСТВА БАЙТ /////////////////
  W25qxx_ReadBytes(buf, 9350, 10);
 /* USER CODE END 2 */
 /* Init code for STM32_WPAN */
MX_APPE_Init();
 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
HAL_ADC_Start(&hadc1); //
 while (1)
 {
        HAL_ADC_PollForConversion(&hadc1, 100);
  /* USER CODE END WHILE */
MX_APPE_Process();
/* USER CODE BEGIN 3 */
if(data_flg==true)//если данные готовы(о чем сообщит прерывание от DRDYB)
        {
                adc = HAL_ADC_GetValue(&hadc1);;//читаем данные с ADS1293 в массив buffer
                //с ads приходит 24 битные данные в количестве 3 штук = 9 байт за один проход
                //на флешку пишем постранично, передавая данные побайтово
                for (int i=0;i<=127;i++){//пока количество снятий показаний меньше 28, новые данные записывваются в буферный массив
                        //там они ждут, пока накопятся достаточное количество для записи на флешпамять(записываем страницу 256 байт)
                        databuffer[0+3*i] = (adc& 0x000000ff);
                        databuffer[1+3*i] = (adc& 0x0000ff00) >> 8;
                        databuffer[2+3*i] = (adc& 0x00ff0000) >> 16;
                        databuffer[3+3*i] = (adc& 0xff000000) >> 24;
                        databuffer[4+3*i] = (adc& 0x000000ff);
                        databuffer[5+3*i] = (adc& 0x0000ff00) >> 8;
                        databuffer[6+3*i] = (adc& 0x00ff0000) >> 16;
                        databuffer[7+3*i] = (adc& 0xff000000) >> 24;
                        databuffer[8+3*i] = (adc& 0x000000ff);
                        databuffer[9+3*i] = (adc& 0x0000ff00) >> 8;
                        databuffer[10+3*i] = (adc& 0x00ff0000) >> 16;
                        databuffer[11+3*i] = (adc& 0xff000000) >> 24;
                        if(i>127){//если превысили число измерений, то
                                i=0;//сбрасываемсчетчик
                                for (int numberPage=0; numberPage<=w25qxx.BlockCount; numberPage=numberPage+256){
                                        //записываем данные на флешку,
                                        //пока numberPage номер страницы меньше чем число страниц на всей флешке
```

```
                                        uint8_t clear = W25qxx_IsEmptyPage(0, 40);//в clear записывается 1, если чистая страница, 0 если занята данными
                                        if(clear==1) W25qxx_WritePage(databuffer, numberPage, 0, 256);//есличистопишем 256 байт
                                            }
                                        }
                                    }
  }
  /* USER CODE END 3 */
  * @brief System Clock Configuration
  * @retval None
void SystemClock_Config(void)
RCC_OscInitTypeDefRCC_OscInitStruct = {0};
RCC_ClkInitTypeDefRCC_ClkInitStruct = {0};
  /** Configure LSE Drive Capability
HAL_PWR_EnableBkUpAccess();
  __HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);
  /** Configure the main internal regulator output voltage
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|RCC_OSCILLATORTYPE_HSE
                    |RCC_OSCILLATORTYPE_LSE;
RCC_OscInitStruct.HSEState = RCC_HSE_ON;
RCC_OscInitStruct.LSEState = RCC_LSE_ON;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV2;
RCC_OscInitStruct.PLL.PLLN = 8;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
Error_Handler();
  /** Configure the SYSCLKSource, HCLK, PCLK1 and PCLK2 clocks dividers
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK4|RCC_CLOCKTYPE_HCLK2
                    |RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                    |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV16;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.AHBCLK2Divider = RCC_SYSCLK_DIV2;
  RCC_ClkInitStruct.AHBCLK4Divider = RCC_SYSCLK_DIV1;
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK)
```

```
  /** Enables the Clock Security System
HAL_RCC_EnableCSS();
  * @brief Peripherals Common Clock Configuration
void PeriphCommonClock_Config(void)
RCC_PeriphCLKInitTypeDefPeriphClkInitStruct = {0};
  /** Initializes the peripherals clock
PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_SMPS|RCC_PERIPHCLK_RFWAKEUP;
PeriphClkInitStruct.RFWakeUpClockSelection = RCC_RFWKPCLKSOURCE_HSE_DIV1024;
PeriphClkInitStruct.SmpsClockSelection = RCC_SMPSCLKSOURCE_HSI;
PeriphClkInitStruct.SmpsDivSelection = RCC_SMPSCLKDIV_RANGE1;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
  /* USER CODE BEGIN Smps */
  /* USER CODE END Smps */
  * @brief ADC1 Initialization Function
  * @param None
static void MX_ADC1_Init(void)
  /* USER CODE BEGIN ADC1_Init 0 */
  /* USER CODE END ADC1_Init 0 */
ADC_ChannelConfTypeDefsConfig = {0};
  /* USER CODE BEGIN ADC1_Init 1 */
  /* USER CODE END ADC1_Init 1 */
  /** Common config
  hadc1.Instance = ADC1;
hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
hadc1.Init.Resolution = ADC_RESOLUTION_12B;
hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
hadc1.Init.LowPowerAutoWait = DISABLE;
hadc1.Init.ContinuousConvMode = DISABLE;
hadc1.Init.NbrOfConversion = 1;
hadc1.Init.DiscontinuousConvMode = DISABLE;
hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIG_T1_TRGO;
hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_RISING;
hadc1.Init.DMAContinuousRequests = DISABLE;
hadc1.Init.Overrun = ADC_OVR_DATA_PRESERVED;
hadc1.Init.OversamplingMode = DISABLE;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  /** Configure Regular Channel
sConfig.Channel = ADC_CHANNEL_1;
sConfig.Rank = ADC_REGULAR_RANK_1;
sConfig.SamplingTime = ADC_SAMPLETIME_2CYCLES_5;
sConfig.SingleDiff = ADC_SINGLE_ENDED;
sConfig.OffsetNumber = ADC_OFFSET_NONE;
sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  /* USER CODE BEGIN ADC1_Init 2 */
```

```
  /* USER CODE END ADC1_Init 2 */
  * @brief I2C1 Initialization Function
static void MX_I2C1_Init(void)
  /* USER CODE BEGIN I2C1_Init 0 */
  /* USER CODE END I2C1_Init 0 */
  /* USER CODE BEGIN I2C1_Init 1 */
  /* USER CODE END I2C1_Init 1 */
  hi2c1.Instance = I2C1;
hi2c1.Init.Timing = 0x00000E14;
hi2c1.Init.OwnAddress1 = 0;
hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
hi2c1.Init.OwnAddress2 = 0;
hi2c1.Init.OwnAddress2Masks = I2C_OA2_NOMASK;
hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  /** Configure Analogue filter
  if (HAL_I2CEx_ConfigAnalogFilter(&hi2c1, I2C_ANALOGFILTER_ENABLE) != HAL_OK)
  /** Configure Digital filter
  if (HAL_I2CEx_ConfigDigitalFilter(&hi2c1, 0) != HAL_OK)
  /* USER CODE BEGIN I2C1_Init 2 */
  /* USER CODE END I2C1_Init 2 */
  * @brief IPCC Initialization Function
static void MX_IPCC_Init(void)
  /* USER CODE BEGIN IPCC_Init 0 */
  /* USER CODE END IPCC_Init 0 */
  /* USER CODE BEGIN IPCC_Init 1 */
  /* USER CODE END IPCC_Init 1 */
hipcc.Instance = IPCC;
  if (HAL_IPCC_Init(&hipcc) != HAL_OK)
  /* USER CODE BEGIN IPCC_Init 2 */
  /* USER CODE END IPCC_Init 2 */
  * @brief RF Initialization Function
static void MX_RF_Init(void)
  /* USER CODE BEGIN RF_Init 0 */
  /* USER CODE END RF_Init 0 */
  /* USER CODE BEGIN RF_Init 1 */
  /* USER CODE END RF_Init 1 */
  /* USER CODE BEGIN RF_Init 2 */
  /* USER CODE END RF_Init 2 */
  * @brief RTC Initialization Function
static void MX_RTC_Init(void)
  /* USER CODE BEGIN RTC_Init 0 */
  /* USER CODE END RTC_Init 0 */
  /* USER CODE BEGIN RTC_Init 1 */
  /* USER CODE END RTC_Init 1 */
```

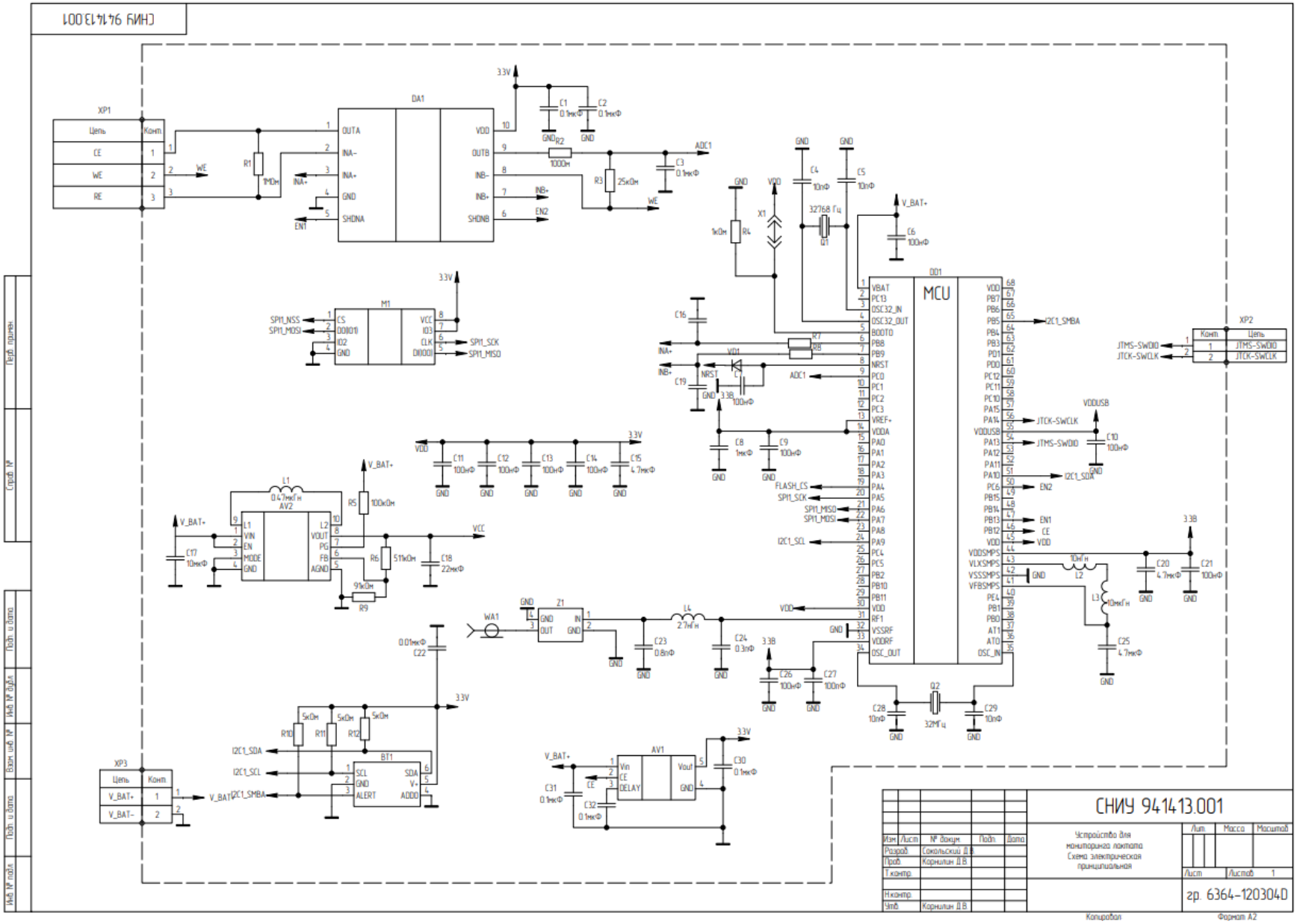
```
  /** Initialize RTC Only
hrtc.Instance = RTC;
hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
hrtc.Init.AsynchPrediv = CFG_RTC_ASYNCH_PRESCALER;
hrtc.Init.SynchPrediv = CFG_RTC_SYNCH_PRESCALER;
hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
hrtc.Init.OutPutRemap = RTC_OUTPUT_REMAP_NONE;
  if (HAL_RTC_Init(&hrtc) != HAL_OK)
  /** Enable the WakeUp
  if (HAL_RTCEx_SetWakeUpTimer_IT(&hrtc, 60, RTC_WAKEUPCLOCK_CK_SPRE_16BITS) != HAL_OK)
  /* USER CODE BEGIN RTC_Init 2 */
  /* USER CODE END RTC_Init 2 */
  * @brief SPI1 Initialization Function
static void MX_SPI1_Init(void)
  /* USER CODE BEGIN SPI1_Init 0 */
  /* USER CODE END SPI1_Init 0 */
  /* USER CODE BEGIN SPI1_Init 1 */
  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
hspi1.Init.Mode = SPI_MODE_MASTER;
hspi1.Init.Direction = SPI_DIRECTION_2LINES;
hspi1.Init.DataSize = SPI_DATASIZE_4BIT;
hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_HARD_OUTPUT;
hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
hspi1.Init.CRCPolynomial = 7;
hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
hspi1.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  /* USER CODE BEGIN SPI1_Init 2 */
  /* USER CODE END SPI1_Init 2 */
  * @brief TIM1 Initialization Function
static void MX_TIM1_Init(void)
  /* USER CODE BEGIN TIM1_Init 0 */
  /* USER CODE END TIM1_Init 0 */
TIM_ClockConfigTypeDefsClockSourceConfig = {0};
TIM_MasterConfigTypeDefsMasterConfig = {0};
  /* USER CODE BEGIN TIM1_Init 1 */
  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
```

```
htim1.Init.Prescaler = 8000-1;
htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
htim1.Init.Period = 65535;
htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim1.Init.RepetitionCounter = 0;
htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterOutputTrigger2 = TIM_TRGO2_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  /* USER CODE BEGIN TIM1_Init 2 */
  /* USER CODE END TIM1_Init 2 */
  * @brief TIM16 Initialization Function
static void MX_TIM16_Init(void)
  /* USER CODE BEGIN TIM16_Init 0 */
  /* USER CODE END TIM16_Init 0 */
TIM_OC_InitTypeDefsConfigOC = {0};
TIM_BreakDeadTimeConfigTypeDefsBreakDeadTimeConfig = {0};
  /* USER CODE BEGIN TIM16_Init 1 */
  /* USER CODE END TIM16_Init 1 */
  htim16.Instance = TIM16;
htim16.Init.Prescaler = 0;
htim16.Init.CounterMode = TIM_COUNTERMODE_UP;
htim16.Init.Period = 65535;
htim16.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim16.Init.RepetitionCounter = 0;
htim16.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim16) != HAL_OK)
  if (HAL_TIM_PWM_Init(&htim16) != HAL_OK)
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = 0;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
  if (HAL_TIM_PWM_ConfigChannel(&htim16, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
sBreakDeadTimeConfig.DeadTime = 0;
sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
sBreakDeadTimeConfig.BreakFilter = 0;
```

```
sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim16, &sBreakDeadTimeConfig) != HAL_OK)
  /* USER CODE BEGIN TIM16_Init 2 */
  /* USER CODE END TIM16_Init 2 */
HAL_TIM_MspPostInit(&htim16);
  * @brief TIM17 Initialization Function
static void MX_TIM17_Init(void)
  /* USER CODE BEGIN TIM17_Init 0 */
  /* USER CODE END TIM17_Init 0 */
  /* USER CODE BEGIN TIM17_Init 1 */
  /* USER CODE END TIM17_Init 1 */
  htim17.Instance = TIM17;
htim17.Init.Prescaler = 0;
htim17.Init.CounterMode = TIM_COUNTERMODE_UP;
htim17.Init.Period = 65535;
htim17.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim17.Init.RepetitionCounter = 0;
htim17.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim17) != HAL_OK)
  if (HAL_TIM_PWM_Init(&htim17) != HAL_OK)
  if (HAL_TIM_PWM_ConfigChannel(&htim17, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim17, &sBreakDeadTimeConfig) != HAL_OK)
  /* USER CODE BEGIN TIM17_Init 2 */
  /* USER CODE END TIM17_Init 2 */
HAL_TIM_MspPostInit(&htim17);
  * @brief USART1 Initialization Function
static void MX_USART1_UART_Init(void)
  /* USER CODE BEGIN USART1_Init 0 */
  /* USER CODE END USART1_Init 0 */
  /* USER CODE BEGIN USART1_Init 1 */
  /* USER CODE END USART1_Init 1 */
  huart1.Instance = USART1;
huart1.Init.BaudRate = 115200;
huart1.Init.WordLength = UART_WORDLENGTH_8B;
huart1.Init.StopBits = UART_STOPBITS_1;
huart1.Init.Parity = UART_PARITY_NONE;
huart1.Init.Mode = UART_MODE_TX_RX;
huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart1.Init.OverSampling = UART_OVERSAMPLING_16;
huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
huart1.Init.ClockPrescaler = UART_PRESCALER_DIV1;
huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  if (HAL_UARTEx_SetTxFifoThreshold(&huart1, UART_TXFIFO_THRESHOLD_1_8) != HAL_OK)
  if (HAL_UARTEx_SetRxFifoThreshold(&huart1, UART_RXFIFO_THRESHOLD_1_8) != HAL_OK)
  if (HAL_UARTEx_DisableFifoMode(&huart1) != HAL_OK)
  /* USER CODE BEGIN USART1_Init 2 */
```

```
  /* USER CODE END USART1_Init 2 */
  * @brief GPIO Initialization Function
static void MX_GPIO_Init(void)
GPIO_InitTypeDefGPIO_InitStruct = {0};
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOB, CE_Pin|EN1_Pin, GPIO_PIN_RESET);
HAL_GPIO_WritePin(EN2_GPIO_Port, EN2_Pin, GPIO_PIN_RESET);
  /*Configure GPIO pins :CE_Pin EN1_Pin */
GPIO_InitStruct.Pin = CE_Pin|EN1_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
  /*Configure GPIO pin : EN2_Pin */
GPIO_InitStruct.Pin = EN2_Pin;
HAL_GPIO_Init(EN2_GPIO_Port, &GPIO_InitStruct);
/* USER CODE BEGIN 4 */
/* USER CODE END 4 */
  * @brief  This function is executed in case of error occurrence.
void Error_Handler(void)
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  /* USER CODE END Error_Handler_Debug */
#ifdef  USE_FULL_ASSERT
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
void assert_failed(uint8_t *file, uint32_t line)
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
#endif /* USE_FULL_ASSERT */
```

ПРИЛОЖЕНИЕ Б



ПРИЛОЖЕНИЕ В

Зона	Поз обозначение	Наименование	Кол.	Примечание
		Регуляторы напряжения		
	AV1	R1517S331D-E2-FE	1	
	AV2	TPS63802DLAR	1	
		Датчики		
	BT1	TMP102AIDRLR	1	
		Конденсаторы		
	C1-C3	GRM55DR73A104KW01L	3	
	C4,C5	GRM2165C1H100JZ01D	2	
	C6,C7	GRM155R61C104KA88D	2	
	C8	GRM55DR72A105KA01L	1	
	C9-C14	GRM155R61C104KA88D	6	
	C15	GRM55ER72A475KA01L	1	
	C16	GRM033R60J153KE01D	1	
	C17	GRM21BR61C106KE15L	1	
	C18	GRM31CR61C226KE15L	1	
	C19	GRM033R60J153KE01D	1	
	C20	GRM55ER72A475KA01L	1	
	C21	GRM155R61C104KA88D	1	
	C22	GRM21BR72E103KW03K	1	
	C23	GRM2165C1HR82CD01D	1	
	C24	GRM1885C1HR30CD01D	1	
	C25	GRM55ER72A475KA01L	1	
	C26	GRM155R61C104KA88D	1	
	C27	GRM2165C1H101JA01D	1	
	C28,C29	GRM2165C1H100JZ01D	2	
	C30...C32	GRM55DR73A104KW01L	3	

Перв. примен.

Справ. №

Подп. и дата

Инв. № дубл.

Взам. инв. №

Подп. и дата

Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Сокольский Д.В.		
Пров.		Корнилин Д.В.		
Н.контр.				
Утв.		Корнилин Д.В.		

СНИУ 941413.001 ПЭ3

Устройство для мониторинга лактата
Перечень элементов

Лит.	Лист	Листов
	1	3

гр. 6364-120304D

Копировал

Формат А4

Зона	Поз. обозначение	Наименование	Кол.	Примечание
		Микросхемы		
	DA1	MAX9913EUB	1	
	DD1	STM32WB55RCV6	1	
		Генераторы		
	G1	32768-HC49U-18-49U	1	
	G2	32.768MHzHC-49U	1	
		Дроссели		
	L1	LQM21NNR47K10D	1	
	L2	LQW15AN10NJ00D	1	
	L3	LQM21FN100N00B	1	
	L4	B82498F3279K000	1	
		Устройства памяти		
	M1	W25Q32	1	
		Резисторы		
	R1	RI0805L9763FT	1	
	R2	RC0402JR-07100RL	1	
	R3	RI0603L2492FT	1	
	R4	RC0402FR-071KL	1	
	R5	RC0402FR-07100KL	1	
	R6	RC0402FR-07510KL	1	
	R7,R8	RC0402FR-07100KL	2	
	R9	RC0402FR-0791KL	1	
	R10..R12	RC0402FR-075K1L	3	
		Диод		
	VD1	1N4934	1	

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Изм	Лист	№ докум.	Подп.	Дата	СНИУ 941413.001 ПЭЗ	Лист 2

Копировал Формат А4

Зона	Поз. обозначение	Наименование	Кол.	Примечание
		Антенны		
	WA1	1909763-1	1	
		Соединители		
	X1	116-CG1	1	
	XP1	24.131.1	1	
	XP2	TC2030-PKT-SWD	1	
	XP3	L-KLS5-CR2032-07	1	
		Фильтры		
	Z1	DLF162500LT-5028A1	1	

Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Изм	Лист	№ докум.	Подп.	Дата	СНИУ 941413.001 ПЭ3	Лист 3

Копировал Формат А4