МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
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 курсовому проекту

”Устройство считывания данных для непрерывного мониторинга уровня глюкозы”

Выполнил студент группы 6364-120304D: _____ Согонов Е.А.

Руководитель проекта: _____ Корнилин Д.В.

Работа защищена с оценкой: _____

Самара 2023

ЗАДАНИЕ

Разработать монитор активности и отслеживания падений со следующими параметрами:

– Тип датчика – потенциометрический

– Диапазон измеряемых сопротивлений от 100 до 10 кОм

– Погрешность измерения сопротивления 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: 34 страниц, 14 рисунков, источников, 1 приложение.

ГЛЮКОЗА, УСТРОЙСТВО СЧИТЫВАНИЯ ДАННЫХ, МИКРОКОНТРОЛЛЕР, BLUETOOTH, STM32WB55RCV6

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

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Диабет превратился в одну из основных эпидемий здравоохранения современной эпохи. Ожидается, что во всем мире общее число людей с диабетом вырастет со 171 миллиона в 2000 году до 366 миллионов в 2030 году. [1]

Мало того, что диабет был шестой по значимости причиной смерти, указанной в свидетельствах о смерти в США в 2020 году, но предполагаемая стоимость диабета в Соединенных Штатах в 2002 году составила 132 миллиарда долларов, включая как прямые, так и косвенные расходы (инвалидность, инвалидность, потеря работы, преждевременная смертность).

Известно, что строгий гликемический контроль снижает разрушительные и дорогостоящие вторичные микро- и макрососудистые осложнения, связанные с диабетом, тем самым улучшая качество жизни миллионов пациентов с диабетом и значительно сокращая расходы на здравоохранение. Также, строгий контроль уровня глюкозы обеспечивает клинические преимущества у пациентов в критическом состоянии.

Непрерывный мониторинг глюкозы(НМГ) – метод регистрации изменений концентрации глюкозы в крови, при котором результаты измерений фиксируются не реже чем каждые 5 мин на протяжении длительного времени (более суток). Применяемые в настоящее время устройства для НМГ позволяют получить данные о гликемии косвенно по концентрации глюкозы в межтканевой жидкости.

В данном курсовом проекте рассматривается способ создания устройства на базе микроконтроллера, который сможет отслеживать уровень глюкозы в крови человека. В процессе были подобраны необходимые в задании микроконтроллер с интегрированным модулем Bluetooth, акселерометр, а также написана управляющая программа на языке Си.

1 РАЗРАБОТКА СТРУКТУРНОЙ СХЕМЫ УСТРОЙСТВА

Структурная схема устройства представлена на рисунке 1.

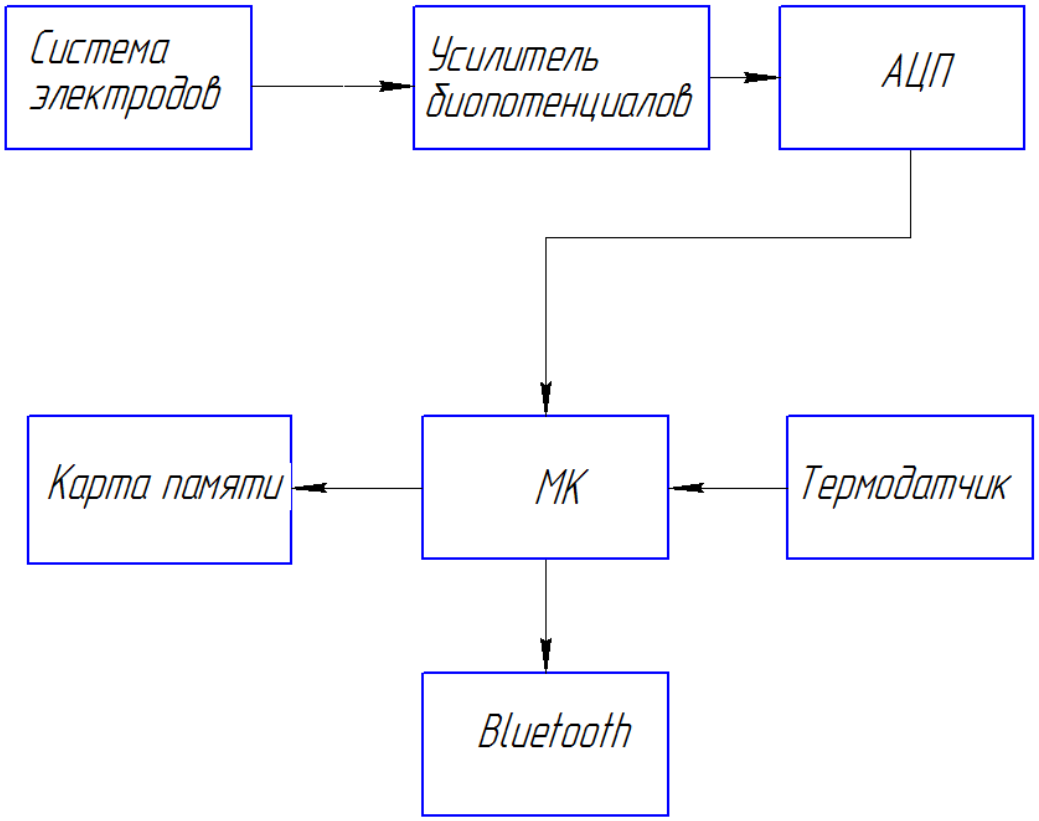


Рисунок 1 – Структурная схема устройства

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 во внешнюю Flash-память.

2 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ СХЕМЫ УСТРОЙСТВА

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 трансимпендансный усилитель на операционном усилителе с чрезвычайно низким током смещения. Его схема показана на рисунке 2.

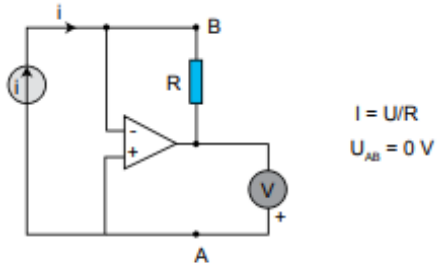


Figure 3—The transimpedance amplifier is a way to automatically adjust the counter voltage. An operational amplifier will set its output in order to have a nearly null voltage offset between its two inputs: U_{AB} = 0, which is exactly what we are looking for.

Рисунок 2 – Трансимпедансный усилитель

С этой целью мы использовали операционный усилитель MAX9913 [2], 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 даташита MAX9913 приведена на рисунке 3.

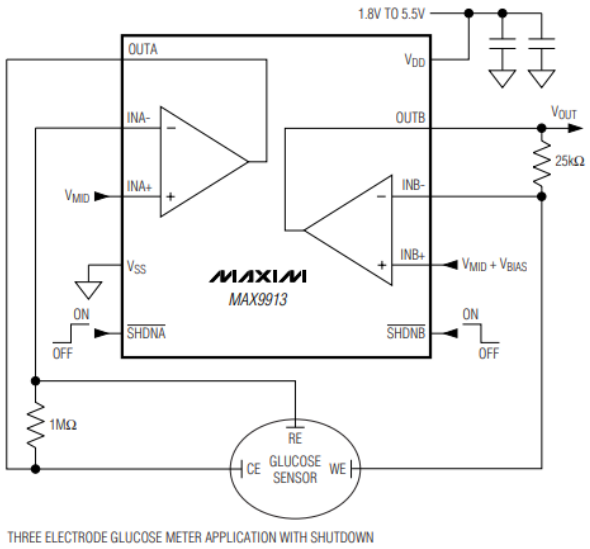


Рисунок 3 – Схема включения MAX9913

Нумерация и назначение выводов MAX9913 приведено ниже (рисунки 4, 5).

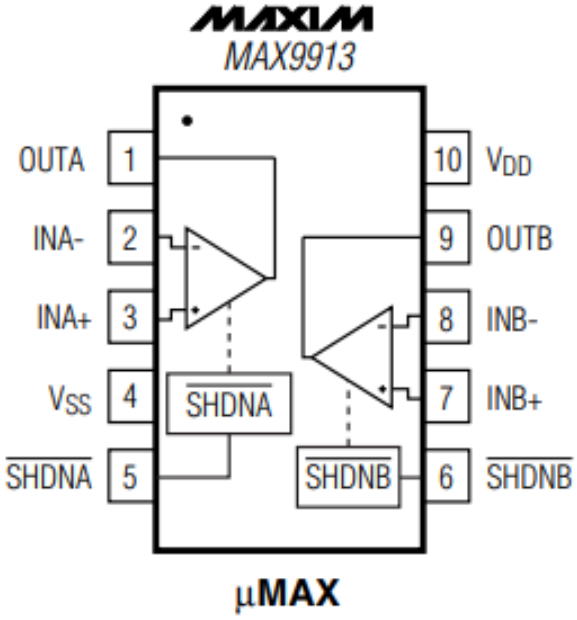


Рисунок 4 – Распиновка MAX9913

Pin Description

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 5 – Назначение выводов MAX9913

2.2 Выбор микроконтроллера

С учетом технического задания микроконтроллер должен обладать следующими свойствами:

– Интерфейс для работы с термодатчиком : I^2C;

– Интерфейс для работы с внешней флеш-памятью: SPI или I^2C;

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

– Малое энергопотребление;

Для решения задачи был выбран микроконтроллер STM32WB55RCV6 фирмы ST Microelectronics [5].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, управляющее радиотрактом и реализующее низкоуровневые функции сетевых протоколов;

Основные характеристики:

– типовое энергопотребление 50 мкА/МГц (при напряжении питания 3 В);

– потребление в режиме останова 1,8 мкА (радиочасть в режиме ожидания (standby));

– потребление в выключенном состоянии (Shutdown) менее 50 нА;

– диапазон допустимых напряжений питания 1,7…3,6 В (встроенный DC-DC–преобразователь и LDO-стабилизатор);

– рабочий температурный диапазон -40…105°C.

Структурная схема микроконтроллера приведена на рисунке 7, а назначение выводов портов корпуса на рисунке 6.

Figure 11. STM32WB55Rx VFQFPN68 pinout(1)(2)

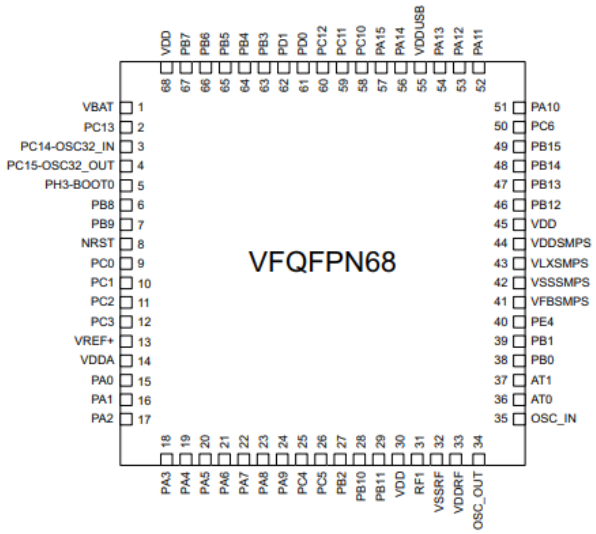


Рисунок 6 – Назначение выводов

Figure 2. STM32WB35xx block diagram

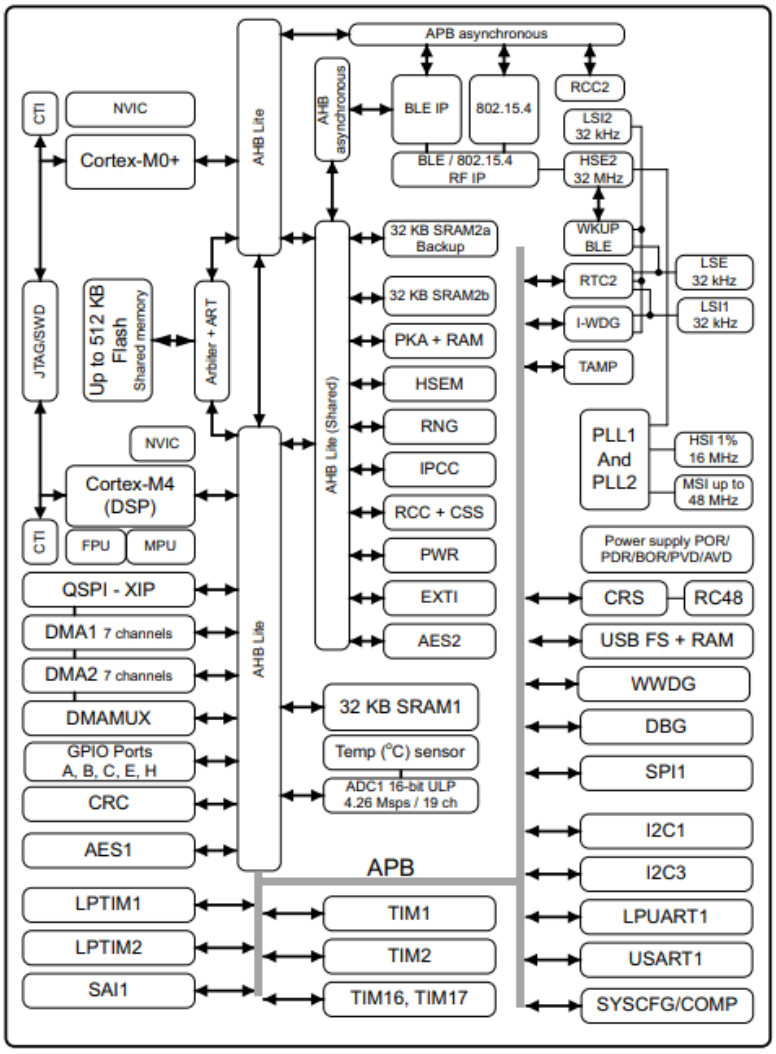


Рисунок 7 – Структурная схема

Подключение будет осуществляться согласно типовой схеме из Application note [6](рисунок 8).

Figure 9. VFQFPN68 reference board

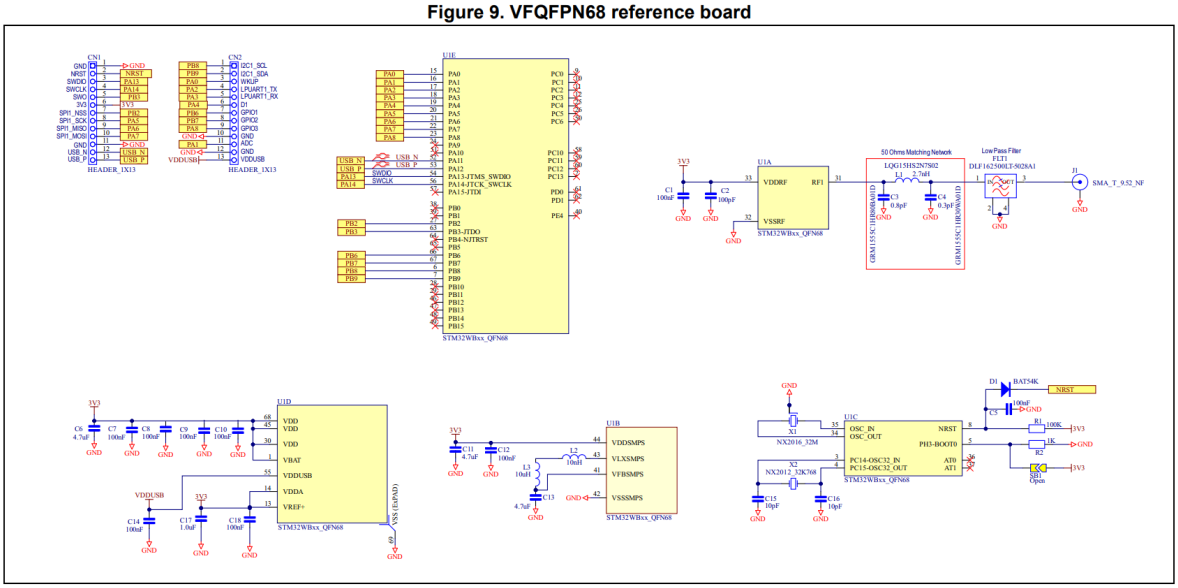


Рисунок 8 – Типовая схема подключения STM32WB55

2.3 Выбор встроенного носителя

В качестве носителя информации выберем последовательную FLASH-память серии W25Q [4]. Данная последовательная память может быть различной ёмкости — 8, 16, 32, 64, 128, 256 Мбит и т. д. Подключается такая память по интерфейсу SPI, а также по многопроводным интерфейсам Dual SPI, Quad SPI и QPI. Мы подключим данную микросхему по обычному интерфейсу SPI. Необходимый объем памяти для наших нужд рассчитывается по формуле: объем памяти= 16*1*60*24 = 23040bit ≈23 Mbit где 16 - необходимый объем памяти для хранения одного измерения в битах, 1- число измерений в минуту, 60 - количество минут в одном часе, 24 - количество часов в сутках. Таким образом, для хранения данных в течение 24 часов нам нужно выбрать FLASH-память с ёмкостью 32 Mbit.

Краткие основные характеристики W25Q:

– Потребляемая мощность и температурный диапазон:

– Напряжение питания 2.7…3.6 В

– Типичный потребляемый ток: 4 мА (активный режим), <1 мкА (в режиме снижения мощности)

– Рабочий температурный диапазон -40°C…+85°C.

Гибкая архитектура с секторами размером 4 кбайт:

– Посекторное стирание (размер каждого сектора 4 кбайт)

– Программирование от 1 до 256 байт

– До 100 тыс. циклов стирания/записи

– 20-летнее хранение данных

Максимальная частота работы микросхемы:

– 104 МГц в режиме SPI

– 208/416 МГц — Dual / Quad SPI

Также микросхема существует в различных корпусах, но в большинстве случаев распространён корпус SMD SO8. Распиновка микросхемы следующая(рисунок 9).

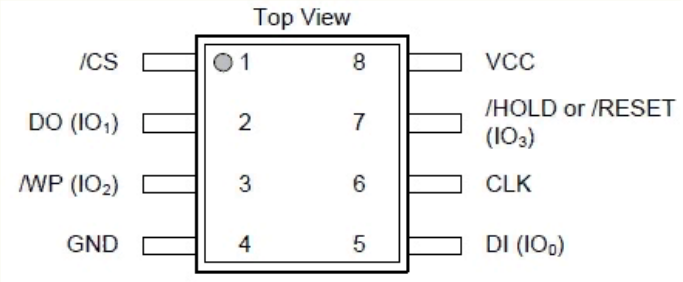


Рисунок 9 – Распиновка W25Q32

Описание выводов из [4](рисунок 10).

3.3 Pin Description SOIC / VSOP 208-mil, WSON 6x5-mm / 8x6-mm

PIN NO.	PIN NAME	I/O	FUNCTION
1	/CS	I	Chip Select Input
2	DO (IO1)	I/O	Data Output (Data Input Output 1)(1)
3	/WP (IO2)	I/O	Write Protect Input (Data Input Output 2)(2)
4	GND		Ground
5	DI (IO0)	I/O	Data Input (Data Input Output 0)(1)
6	CLK	I	Serial Clock Input
7	/HOLD or /RESET (IO3)	I/O	Hold or Reset Input (Data Input Output 3)(2)
8	VCC		Power Supply

Рисунок 10 – Описание выводов W25Q32

К микроконтроллеру подключается по стандартному интерфейсу SPI.

2.4 Выбор встроенного носителя

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 Texas Instruments TMP102 [3]. Типовая схема включения и конфигурация выводов изображены на рисунках 11 и 12. Информацию с него будем получать по шине I^2C

Также микросхема существует в различных корпусах, но в большинстве случаев распространён корпус SMD SO8.

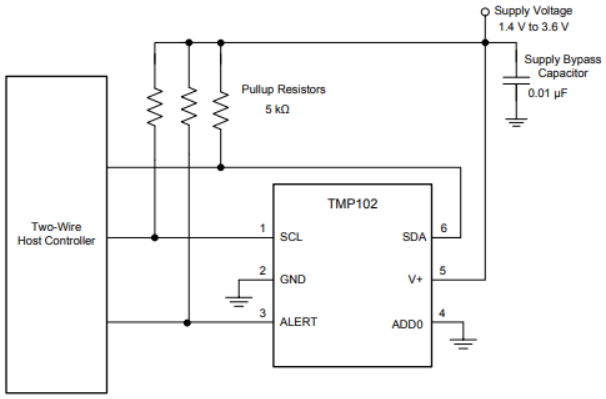
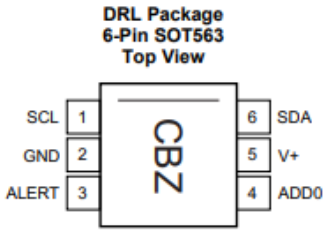


Figure 14. Typical Connections

Рисунок 11 – Типичная схема включения 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 12 – Описание выводов и распиновка TMP102

К микроконтроллеру подключается по стандартному интерфейсу I^2C.

2.5 Схема питания

Питание данного устройства спроектировано от часовой батарейки с напряжением питания 1.5В. В связи с тем, что практически все элементы схемы требуют напряжение питания больше 3 вольт, используем повышающий DC-DC преобразователь напряжения. Возьмем модель от Texas Instruments TPS63802 2-A [7]. Конфигурация выводов изображена на рисунке 13. Данный конвертер поднимет напряжение с 1.8 вольт до 3.5.

3.3 Pin Description SOIC / VSOP 208-mil, WSON 6x5-mm / 8x6-mm

PIN NO.	PIN NAME	I/O	FUNCTION
1	/CS	I	Chip Select Input
2	DO (IO1)	I/O	Data Output (Data Input Output 1)(1)
3	/WP (IO2)	I/O	Write Protect Input (Data Input Output 2)(2)
4	GND		Ground
5	DI (IO0)	I/O	Data Input (Data Input Output 0)(1)
6	CLK	I	Serial Clock Input
7	/HOLD or /RESET (IO3)	I/O	Hold or Reset Input (Data Input Output 3)(2)
8	VCC		Power Supply

Рисунок 13 – Распиновка и описание выводов TPS63802

В целях обеспечения сохранности первоначального вида сигнала, поставим LOW-DROP-OUT регулятор напряжения перед питанием усилителя биопотенциалов.

Возьмем R1517S331D-E2-FE от компании RICOH[3]. На рисунке 14 показано типичное включение данного регулятора в цепь. Данный регулятор обезопасит усилитель от дребезгов питания.

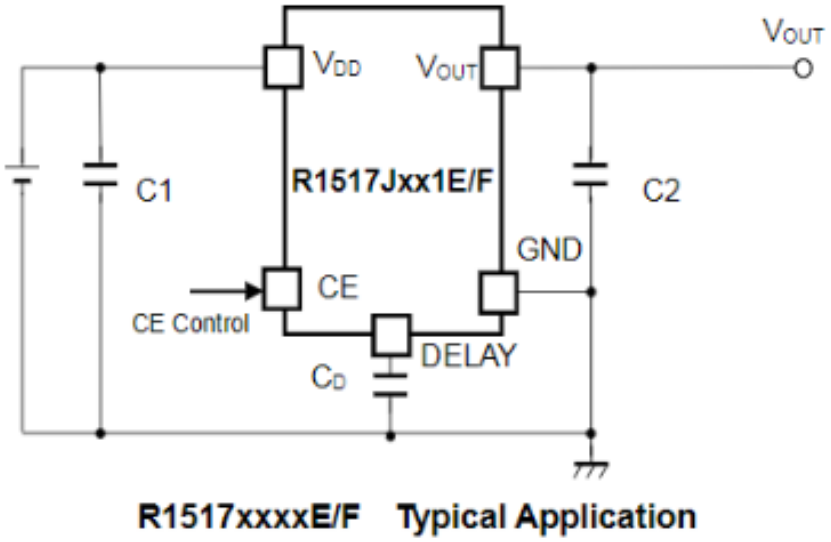


Рисунок 14 – Типовая схема включения R1517S331D

3 РАЗРАБОТКА ПРОГРАММЫ

Алгоритм начинается с включения питания и инициализации всех периферийных устройств и модулей. запуск тактирования, инициализация портов ввода-вывода, таймеров, аналого-цифрового преобразователя, инициализация интерфейсов SPI и I^2C, а так же периферия, необходимая стеку Bluetooth. Далее инициализируется модули, подключенные к микроконтроллеру - термодатчик, flash память и Bluetooth.

После этого начинается основная программа. по таймеру запускается АЦП, получающий сигнал с усилителя биомпедансов. По готовности данных срабатывает прерывание, в обработчике которого происходит запись данных из регистров АЦП в буферный массив. При наполенении массива происходит запись страницы памяти на FLASH память. Массив данных передается по Bluetooth и записывается на карту памяти. Основная программа возвращается на начало своего цикла.

ЗАКЛЮЧЕНИЕ

В данном курсовом проекте рассмотрены принципы разработки устройств на базе микроконтроллеров. Был разработано устройство для непрерывного мониторинга уровня глюкозы. Данные передаются по интерфейсу Bluetooth и записываются на карту памяти. В процессе работы были разработаны структурная и принципиальная схемы устройства, были проведены необходимые расчёты для получения заданной погрешности, осуществлен выбор микроконтроллера и вспомогательных компонентов схемы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Wild S, Roglic G, Green A, Sicree R, King H. Global prevalence of diabetes: estimates for the year 2000 and projections for 2030. Diabetes Care. 2004;27(5):1047-53.

2 Data Sheet на операционный усилитель MAX9913 [Электронный ресурс]. URL:https://doc.softelectronics.ru/docs/op3/MAX9913.pdf(Дата обращения: 11.05.2023)

3 Data Sheet на термодатчик TMP102[Электронный ресурс]. URL:https://dlnmh9ip6v2uc.cloudfront.net/assets/2/6/8/0/c/tmp102-Datasheet.pdf(Дата обращения: 11.05.2023)

4 Data Sheet на последовательную FLASH память W25Q128FV [Электронный ресурс]. URL:https://www.winbond.com/resource-files/W25Q32JV%20RevI%2005042021%20Plus.pdf(Дата обращения: 2.05.2023)

5 Data Sheet на микроконтроллер STM32WB55CCU6 [Электронный ресурс]. URL:https://www.st.com/resource/en/datasheet/stm32wb55cc.pdf (Дата обращения: 10.05.2023)

6 Application note на микроконтроллеры серии STM32WB [Электронный ресурс]. URL:https://www.st.com/resource/en/application_note/an5165-development-of-rf-hardware-using-stm32wb-microcontrollers-stmicroelectronics.pdf (Дата обращения: 11.05.2023)

7 Datasheet на высокоэффективный DC-DC преобразователь TPS63802 [Электронный ресурс]. URL:https://ru.mouser.com/datasheet/2/405/1/tps63802-2403082.pdf (Дата обращения: 11.05.2023)

ПРИЛОЖЕНИЕ А

Основная программа

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2023 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* 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_HandleTypeDef hipcc;
RTC_HandleTypeDef hrtc;
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 страница памяти
uint32_t buffer[3]; //массив для одного измерения
uint8_t send_ble[2]={0,0};
bool data_flg = false;
/* USER CODE END PV */
```

```c
/* 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_TIM16_Init(void);
static void MX_TIM17_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_SPI1_Init(void);
static void MX_TIM1_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_TIM16_Init();
  MX_TIM17_Init();
  MX_USART1_UART_Init();
  MX_SPI1_Init();
  MX_TIM1_Init();
  /* USER CODE BEGIN 2 */
  W25qxx_Init();
  HAL_ADCEx_Calibration_Start(&hadc1, adc);
  HAL_ADC_Start_IT(&hadc1);
  HAL_TIM_Base_Start_IT(&htim1);
  /* 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 записываются в буф
                      //там они ждут, пока накопятся достаточное количество для записи на флеш память(записываем ст
                      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;//сбрасываем счетчик
                            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
                                   if(clear==1) W25qxx_WritePage(databuffer, numberPage, 0, 256);//если чисто пишем 256 байт
                                        }
                                  }
                            }}
   }
    /* USER CODE END 3 */
  }
  /**
```

```
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_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)
  {
    Error_Handler();
  }
  /** Enables the Clock Security System
  */
  HAL_RCC_EnableCSS();
}
/**
  * @brief Peripherals Common Clock Configuration
  * @retval None
  */
void PeriphCommonClock_Config(void)
{
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {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)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN Smps */
  /* USER CODE END Smps */
}
/**
  * @brief ADC1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_ADC1_Init(void)
{
  /* USER CODE BEGIN ADC1_Init 0 */
  /* USER CODE END ADC1_Init 0 */
  ADC_ChannelConfTypeDef sConfig = {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)
  {
    Error_Handler();
  }
  /** 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)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC1_Init 2 */
  /* USER CODE END ADC1_Init 2 */
```

```
}
/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
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)
  {
    Error_Handler();
  }
  /** Configure Analogue filter
  */
  if (HAL_I2CEx_ConfigAnalogFilter(&hi2c1, I2C_ANALOGFILTER_ENABLE) != HAL_OK)
  {
    Error_Handler();
  }
  /** Configure Digital filter
  */
  if (HAL_I2CEx_ConfigDigitalFilter(&hi2c1, 0) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */
  /* USER CODE END I2C1_Init 2 */
}
/**
  * @brief IPCC Initialization Function
  * @param None
  * @retval None
  */
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)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN IPCC_Init 2 */
  /* USER CODE END IPCC_Init 2 */
}
/**
```

```
  * @brief RF Initialization Function
  * @param None
  * @retval None
  */
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
  * @param None
  * @retval None
  */
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)
  {
    Error_Handler();
  }
  /** Enable the WakeUp
  */
  if (HAL_RTCEx_SetWakeUpTimer_IT(&hrtc, 60, RTC_WAKEUPCLOCK_CK_SPRE_16BITS) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN RTC_Init 2 */
  /* USER CODE END RTC_Init 2 */
}
/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
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)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */
  /* USER CODE END SPI1_Init 2 */
}
/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{
  /* USER CODE BEGIN TIM1_Init 0 */
  /* USER CODE END TIM1_Init 0 */
  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  /* USER CODE BEGIN TIM1_Init 1 */
  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 0;
  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)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterOutputTrigger2 = TIM_TRGO2_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */
  /* USER CODE END TIM1_Init 2 */
}
/**
```

```
  * @brief TIM16 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM16_Init(void)
{
  /* USER CODE BEGIN TIM16_Init 0 */
  /* USER CODE END TIM16_Init 0 */
  TIM_OC_InitTypeDef sConfigOC = {0};
  TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {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)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim16) != HAL_OK)
  {
    Error_Handler();
  }
  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)
  {
    Error_Handler();
  }
  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)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM16_Init 2 */
  /* USER CODE END TIM16_Init 2 */
  HAL_TIM_MspPostInit(&htim16);
}
/**
  * @brief TIM17 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM17_Init(void)
```

```
{
  /* USER CODE BEGIN TIM17_Init 0 */
  /* USER CODE END TIM17_Init 0 */
  TIM_OC_InitTypeDef sConfigOC = {0};
  TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {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)
  {
    Error_Handler();
  }
  if (HAL_TIM_PWM_Init(&htim17) != HAL_OK)
  {
    Error_Handler();
  }
  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(&htim17, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
  {
    Error_Handler();
  }
  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(&htim17, &sBreakDeadTimeConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM17_Init 2 */
  /* USER CODE END TIM17_Init 2 */
  HAL_TIM_MspPostInit(&htim17);
}
/**
  * @brief USART1 Initialization Function
  * @param None
  * @retval None
  */
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)
  {
    Error_Handler();
  }
  if (HAL_UARTEx_SetTxFifoThreshold(&huart1, UART_TXFIFO_THRESHOLD_1_8) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_UARTEx_SetRxFifoThreshold(&huart1, UART_RXFIFO_THRESHOLD_1_8) != HAL_OK)
  {
    Error_Handler();
  }
  if (HAL_UARTEx_DisableFifoMode(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART1_Init 2 */
  /* USER CODE END USART1_Init 2 */
}
/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_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);
  /*Configure GPIO pin Output Level */
  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;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  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.
  * @retval None
  */
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();
  while (1)
  {
  }
  /* 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
  * @retval None
  */
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 */
```