Министерство науки и высшего образования Российской Федерации
Филиал федерального государственного автономного образовательного учреждения высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)» в г. Миассе
Факультет «Электротехнический»
Кафедра «Автоматика»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_____С.С. Голощапов
_____2019 г.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УЧЕТА ТЕПЛОВОЙ ЭНЕРГИИ НА ОСНОВЕ ЭНЕРГОЭФФЕКТИВНЫХ СЕТЕЙ ДАЛЬНЕГО РАДИУСА ДЕЙСТВИЯ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИЦИОННОЙ РАБОТЕ
ЮУрГУ – 27.03.04.2019.374 ПЗ ВКР

Руководитель работы
ст. преп. кафедры АиУ
_____ Е.А. Канашев
_____2019 г.

Автор проекта
студент группы МиЭт-468
_____С.В. Антонов
_____2019 г.

Нормоконтролер
кафедры АиУ
_____Т.А. Барбасова
_____2019 г.

Миасс 2019

АННОТАЦИЯ

Антонов С.В. Автоматизированная система учета тепловой энергии на основе энергоэффективных сетей дальнего радиуса действия. – Миасс: ЮУрГУ, МиЭт; 2019, 80 с., 20 ил., 20 таблиц, библиогр. список – 70 наим., 3 прил, 4 листа чертежей ф. А4.

В данной работе рассматриваются задачи проектирования автоматизированной системы учета тепловой энергии на основе энергоэффективных сетей дальнего радиуса действия.

В первом разделе работы описываются существующие решения в области учета тепловой энергии, рассматриваются основные достоинства и недостатки систем учета.

Во втором разделе работы производится анализ энергоэффективных технологий беспроводной передачи данных. В результате анализа для построения автоматизированной системы учета выбрана технология LoRaWAN.

Третий раздел работы посвящен описанию разрабатываемой системы учета тепловой энергии.

Описание выбора элементов, на основе которых проектируется система учета представлено в четвертом разделе работы.

В пятом разделе представлено описание устройства сбора и передачи данных.

Для устройства сбора и передачи данных были разработаны библиотеки на языке программирования С. Функции, описанные в данных библиотеках, используются при создании основной программы УСПД.

В конце пятого раздела приводится описание прототипа устройства сбора и передачи данных.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

По оценкам Росстата в 2016 году в России было произведено 1284 млн. Гкал тепловой энергии [1, с. 35]. Значительная часть выработанной тепловой энергии потребляется промышленным производством 42,6 %. Около 31,2% тепловой энергии потребляется населением (домашними хозяйствами). Совокупное потребление тепла сельским хозяйством, строительством, транспортом и прочими видами экономической деятельности составляет 17,3%. Оставшиеся 8,9% тепловой энергии являются потерями в тепловых сетях [1, с. 41].

Рассмотрим основные элементы, которые входят в структуру тепловых сетей, назначение данных элементов, а также, каким образом осуществляется контроль состояния тепловых сетей и учет потребленной тепловой энергии.

Источниками тепловой энергии в настоящий момент времени являются теплоэлектроцентрали (ТЭЦ), районные и квартирные котельные. От источников тепловая энергия передается по тепловым сетям конечным потребителям энергии. Носителем тепловой энергии является специально подготовленная вода либо пар.

Согласно федеральному закону "О теплоснабжении" под тепловыми сетями понимают совокупность устройств, предназначенных для передачи тепловой энергии, теплоносителя от источников тепловой энергии до теплопотребляющих установок [2].

На рисунке 1 представлена обобщенная структура тепловых сетей.

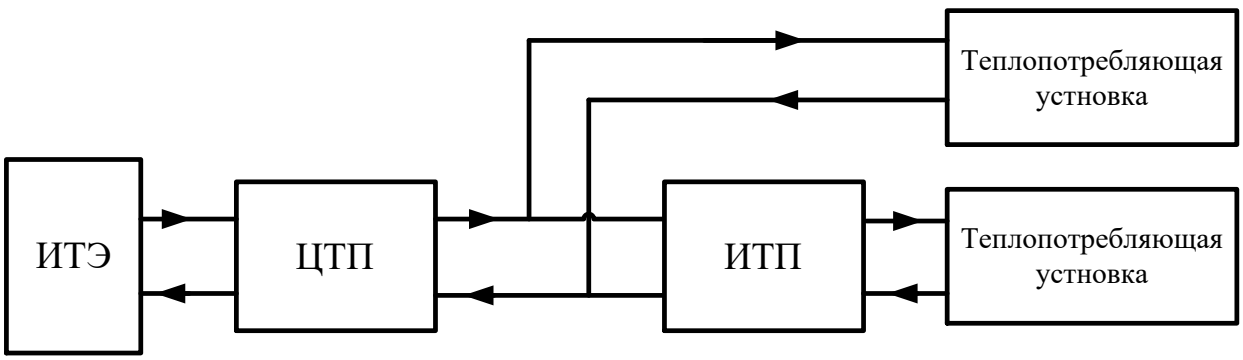

Рисунок 1 – Обобщенная структура тепловой сети

Обозначения, представленные на рисунке 1: ИТЭ – источник тепловой энергии; ЦТП – центральный тепловой пункт; ИТП – индивидуальный тепловой пункт.

В состав тепловых сетей входят следующие элементы:

– центральные тепловые пункты;

– индивидуальные тепловые пункты;

– насосные станции;

– магистральные трубопроводы, распределительные сети.

Центральные тепловые пункты преобразуют термодинамические параметры теплоносителя (давление и температуру) магистрального трубопровода к требуемым параметрам теплоносителя для распределительных сетей, идущих к потребителям. В большинстве случаев центральные тепловые пункты обслуживают несколько потребителей тепловой энергии. В состав центральных тепловых пунктов могут входить насосные и водонагревательные установки.

Узлы присоединения потребителей тепла к тепловым сетям называют абонентскими вводами. Если абонентский ввод сооружается для отдельной технологической установки, здания, то его называют индивидуальным тепловым пунктом [3, с. 26].

Основным назначением насосных станций является увеличение напора в подающем трубопроводе, повышение расхода теплоносителя.

Теплопотребляющими установками являются системы отопления, вентиляции, горячего водоснабжения, различные технологические процессы. Теплопотребляющая установка может подключаться непосредственно к магистральным сетям либо через индивидуальные тепловые пункты, что отражено на рисунке 1.

При производстве, передаче и потреблении тепловой энергии производится учет энергии и параметров теплоносителя (давления, температуры). Учет тепловой энергии необходим для расчетов между потребителями и теплоснабжающими организациями. Учет параметров теплоносителя реализуется для расчета потреб-

ленной тепловой энергии, а также для контроля гидравлических режимов тепловых сетей.

Различают коммерческий и технический учет параметров теплоносителя.

Согласно методике осуществления коммерческого учета тепловой энергии, теплоносителя [4] точки учета тепловой энергии должны располагаться:

– на каждом выводе тепловой сети от источника теплоснабжения;

– в точках передачи теплоносителя в смежные тепловые сети или смежным организациям;

– в точках ввода тепловой сети на объекты, где происходит преобразование тепловой энергии (ЦТП, ИТП);

– в точках ввода тепловой энергии непосредственным потребителям.

Коммерческий учет тепловой энергии, теплоносителя организуется в следующих целях [5]:

– осуществления расчетов между теплоснабжающими, теплосетевыми организациями и потребителями тепловой энергии;

– контроля за тепловыми и гидравлическими режимами работы систем теплоснабжения и теплопотребляющих установок;

– контроля за рациональным использованием тепловой энергии, теплоносителя;

– документирования параметров теплоносителя – массы (объема), температуры и давления.

Технический учет параметров теплоносителя предназначен для контроля состояния тепловой сети, обнаружения утечек теплоносителя.

В данной работе рассматриваются задачи проектирования автоматизированной системы коммерческого учета тепловой энергии.

1 ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ В ОБЛАСТИ УЧЕТА ТЕПЛОВОЙ ЭНЕРГИИ

1.1 Общая структура автоматизированной системы учета тепловой энергии

Рассмотрим существующие структуры автоматизированных систем коммерческого учета тепловой энергии. Основными уровнями системы учета тепловой энергии, как и других автоматизированных системах, являются уровни сбора, передачи и представления данных.

На уровне сбора данных происходит сбор первичной информации о следующих параметрах теплоносителя: температуре, давлении и расходе.

Для измерения параметров теплоносителя используются следующие средства измерения:

– измерительные преобразователи температуры;

– измерительные преобразователи давления;

– расходомеры.

В системах учета тепловой энергии используются различные типы расходомеров, но чаще всего применяют механические и электромагнитные расходомеры, ввиду их дешевизны.

Данные о параметрах теплоносителя поступают в тепловычислитель, где происходит расчет количества потребленной тепловой энергии в соответствии с правилами учета тепловой энергии.

В состав тепловычислителя входит один или несколько интерфейсов связи (RS-232, RS-485, CAN, M-Bus, радиоинтерфейс), по которым передаются данные, запрашиваемые диспетчерской системой.

Данные с тепловычислителя поступают на сервер сбора, обработки и хранения данных. С сервера сбора данных, информация о потребленной тепловой энергии передается на АРМ операторов, руководящего персонала.

Необходимо также отметить, что между тепловычислителем и сервером сбора данных присутствуют различные конвертеры интерфейсов. При значительном удалении тепловычислителей от диспетчерского центра используются беспроводные каналы передачи данных на основе GSM/GPRS.

На рисунке 1.1 представлена типовая структура автоматизированной системы учета тепловой энергии.

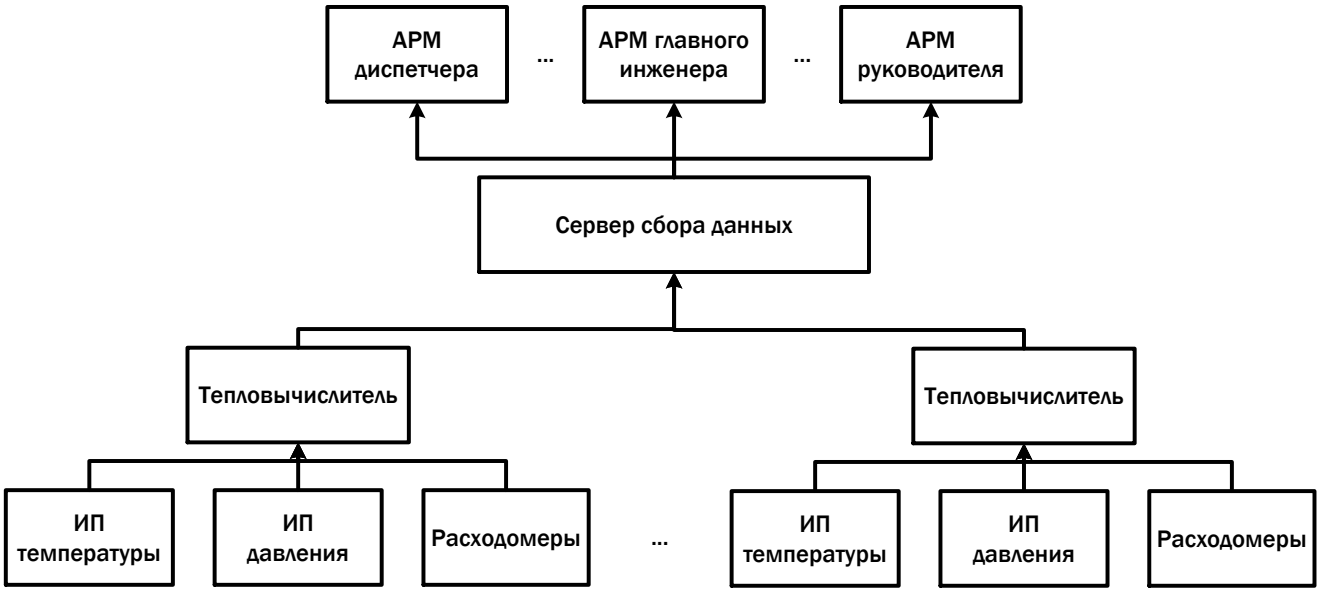

Рисунок 1.1 – Типовая структура автоматизированной системы учета тепловой энергии

Обозначения, принятые на рисунке 1.1: ИП – измерительный преобразователь, АРМ – автоматизированное рабочее место.

1.2 Система учета энергоресурсов на основе сети сотовой связи GSM

Рассмотрим конкретный пример автоматизированной системы учета энергоресурсов, созданной предприятием «МНПП САТУРН» [6]. В состав данной системы входит подсистема учета тепловой энергии.

На рисунке 1.2 представлена структура автоматизированной системы подомового учета тепловой энергии.

Тепловычислители по интерфейсам RS-232, RS-485 подключаются к контроллеру БКД-ПК-RF, который выполняет считывание текущих и архивных данных [7]. Кроме этого БКД-ПК-RF осуществляет контроль состояния бесперебойного источника питания и передает данные посредством внешней GSM антенны.

Сбор показаний осуществляется по сети сотовой связи GSM и Интернет.

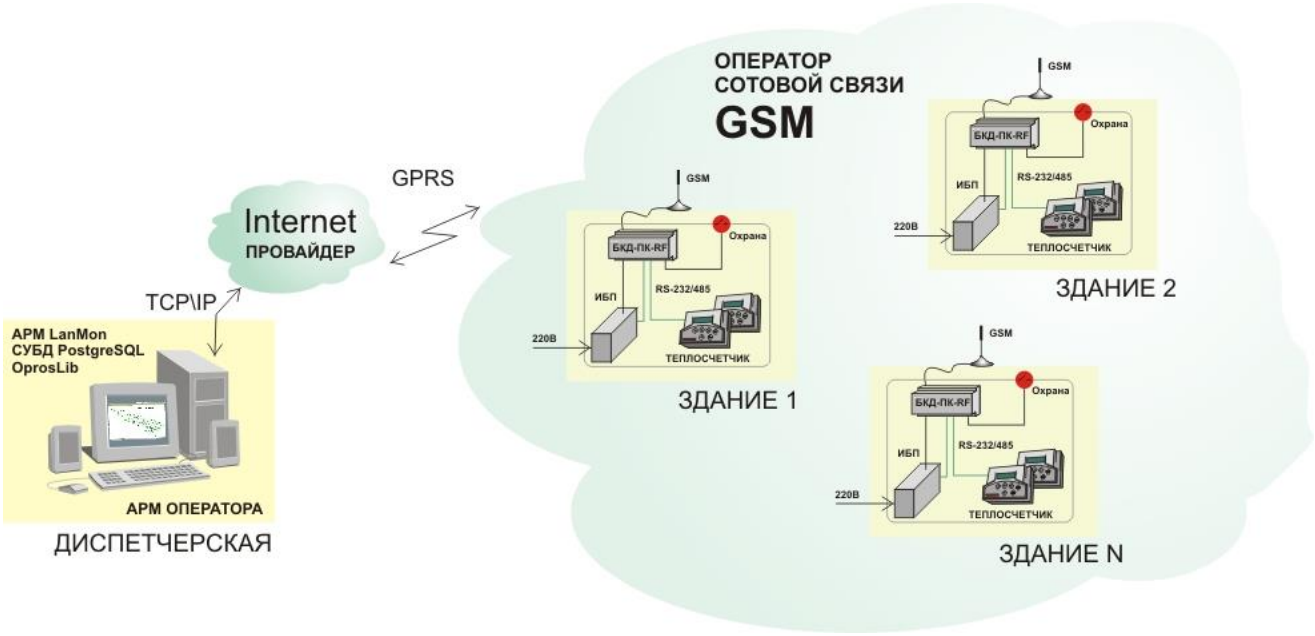

Рисунок 1.2 – Структура автоматизированной системы учета тепловой энергии теплоносителя компании «МНПП САТУРН»

На АРМ оператора устанавливается SCADA-система LanMon, которая предназначена для диспетчеризации приборов учета. Основными функциями данной SCADA-системы являются:

– отображение показаний приборов учета в табличной форме и в виде графиков;

– формирование отчетов по потреблению газа, воды, тепла и электроэнергии;

– формирование тревожных сообщений при отказе тепловычислителей, выходе термодинамических показателей теплоносителя за уставки;

– идентификация персонала;

– ведение журнала событий.

Основными преимуществами представленной автоматизированной системы являются:

– простота получения архивных показаний тепловычислителя;

– удаленный мониторинг основных параметров теплоносителя;

– использование уже существующей сети сотовой связи;

– единый инструмент считывания показаний различных моделей теплосчетчиков.

– отсутствие месячных обходов помещений;

– снижение численности персонала;

– устранение «человеческого фактора» и ошибок.

К недостаткам данной системы учета, основанной на сети GSM, относятся:

– оплата сообщений, передаваемых системой верхнего уровня контроллеру БКД-ПК-RF, а также сообщений, идущих от БКД-ПК-RF на верхний уровень;

– требуется наличие электросети для питания элементов теплосчетчика и модема.

Часто при размещении узлов учета требуется решать вопросы, связанные с организацией электропитания. Поэтому в удаленных от электросети узлах учета устанавливают автономные элементы теплосчетчика. В таких случаях использование модемов GSM приводит к увеличению эксплуатационных расходов, в связи с частой заменой аккумуляторов.

1.3 Требования, предъявляемые к системе коммерческого учета тепловой энергии

Рассматриваемые требования складываются из требований к отдельным элементам системы.

Узел учета тепловой энергии должен оснащаться теплосчетчиками, которые внесены в федеральный информационный фонд по обеспечению единства измерений.

Тепловычислители должны иметь стандартные промышленные интерфейсы и протоколы обмена.

Согласно правилам коммерческого учета тепловой энергии вычислитель теплосчетчика должен иметь нестираемый архив, в который заносятся технические характеристики и коэффициенты прибора [5].

Требования к проектированию узла учета тепловой энергии описаны в правилах коммерческого учета в разделе «Проектирование узлов учета» [5].

Параметры теплоносителя, измерение которых необходимо для осуществления коммерческого учета описаны в правилах учета в разделе «Характеристики тепловой энергии, теплоносителя, подлежащие измерению в целях их коммерческого учета и контроля качества теплоснабжения».

Также следует отметить, что в правилах коммерческого учета тепловой энергии нет требований, предъявляемых к системе учета в целом.

1.4 Выводы по разделу 1

В рамках данного раздела рассмотрена обобщенная структура автоматизированной системы учета тепловой энергии, а также конкретный пример автоматизированной системы учета тепловой энергии, построенной на основе сети сотовой связи GSM. Проанализированы основные требования, предъявляемые к системам коммерческого учета тепловой энергии.

Для решения проблем систем учета на основе сетей GSM, описанных в пункте 1.2, произведем анализ энергоэффективных беспроводных технологий передачи данных. На основании данного анализа выберем наиболее подходящую технологию передачи данных для построения системы учета тепловой энергии.

2 АНАЛИЗ ТЕХНОЛОГИЙ БЕСПРОВОДНОЙ ПЕРЕДАЧИ ДАННЫХ LPWAN

LPWAN (Low-power Wide-Area Network – «энергоэффективные сети дальнего радиуса действия») – беспроводная технология передачи небольших по объему данных на дальние расстояния, разработанная для распределенных сетей телеметрии, межмашинного взаимодействия и интернета вещей [8].

Сети LPWAN могут быть построены на различных технологиях беспроводной передачи данных. Различают сотовые и несотовые стандарты LPWAN. К сотовым стандартам относятся такие технологии беспроводной передачи данных, как: LTE-M, NB-IoT, EC-GSM-IOT. К несотовым стандартам относятся: LoRaWAN, Sigfox, Wheitghless [9]. Рассмотрим достоинства и недостатки части представленных стандартов.

LoRaWAN – это сетевой протокол, разработанный для беспроводного подключения устройств с батарейным питанием к Интернету в региональных, национальных или глобальных сетях и предназначенный для ключевых требований Интернета вещей (IoT), таких как двунаправленные услуги связи, сквозная безопасность, мобильность и локализация [52].

Типовая архитектура сетей построенных на использовании LoRaWAN представлена на рисунке 2.1 [10].

Конечные устройства производят сбор данных о каком-либо требуемом технологическом процессе и передают эти данные на базовые станции при помощи беспроводной технологии передачи LoRa [11]. С базовых станций данные поступают на сетевой сервер, который адресно управляет конечными устройствами, шлюзами сети и соединяет сеть LoRaWAN с серверами приложений.

Конечными устройствами сети LoRaWAN чаще всего являются датчики, либо управляющие устройства.

Сообщения, передаваемые в сети LoRaWAN, подразделяются на два типа: uplink и downlink сообщения. Uplink сообщения – это сообщения, передаваемые от

конечного устройства на сетевой сервер и ретранслируемые через одну или несколько базовых станций. Downlink сообщения передаются от сетевого сервера одному конечному устройству и ретранслируются только одной базовой станцией [53, с. 12].

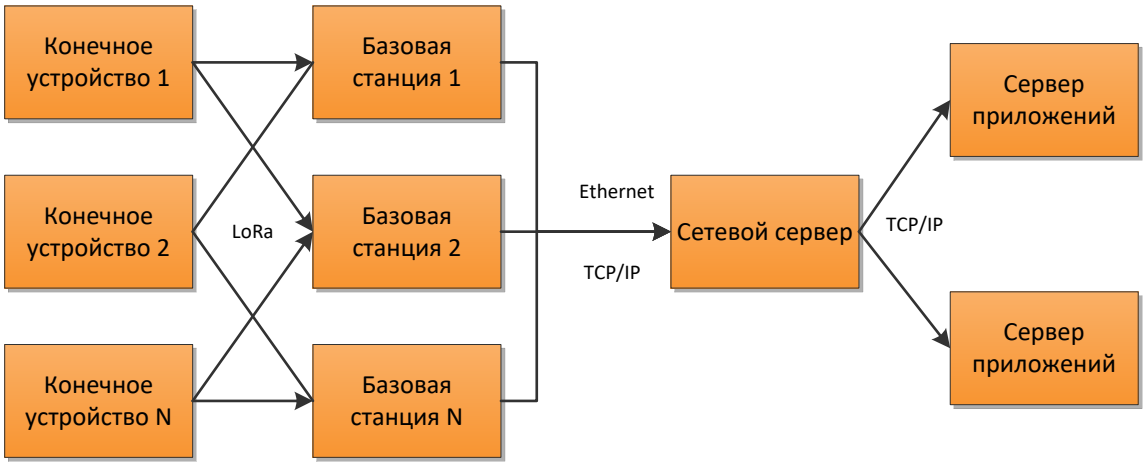

Рисунок 2.1 – Архитектура сетей LoRaWAN

Стандарт LoRaWAN определяет три класса конечных устройств: класс A, B и C. Рассмотрим основные особенности устройств данных классов.

Класса A по умолчанию должен поддерживаться всеми конечными устройствами сети LoRaWAN. Конечное устройство, принадлежащее к данному классу, всегда инициирует передачу данных на базовую станцию согласно собственному расписанию передачи. После передачи сообщения конечное устройство класса A открывает два непродолжительных окна приема подтверждения сообщения, если переданное сообщение было сообщением, на которое сервер должен выслать квитанцию. После получения подтверждения конечное устройство переходит в энергосберегающий режим [12, с. 23].

Конечные устройства класса B обладают свойствами устройств класса A. Дополнительно устройства класса B синхронизируют собственное время и время сети при получении сообщений «маяков» от базовой станции. Синхронизация времени необходима для открытия окон приема конечного устройства в соответствии с графиком сетевого сервера. Это позволяет серверу передавать необходимые

downlink сообщения независимо от графика передачи конечного устройства. В связи с наличием дополнительных окон приема энергопотребление устройств класса B выше, чем у устройств класса A [12, с. 23].

Устройства класса C в дополнении к свойствам устройств класса A поддерживают постоянно открытые окна приема, кроме кратковременных моментов передачи данных. Поэтому сетевой сервер может передать downlink сообщение практически в любой момент времени. Для поддержания приема устройству класса C необходимо постоянное электропитание [12, с. 24].

Основные характеристики сетей LoRaWAN представлены в таблице 2.1.

Таблица 2.1 – Технические характеристики сетей LoRaWAN

Характеристики	Значение	
Тип используемой модуляции	LoRa	
Ширина полосы, кГц	125	
Разделение каналов	CDMA, TDMA	
Скорость передачи данных, кбит/с	от 0,3 до 50	
Дальность связи, км	В городских условиях	В открытой местности
	от 1 до 3	от 15 до 20
Алгоритм шифрования	AES-64 и 128 бит	
Тип частотного диапазона	Нелицензированный (ISM band)	
Частотные диапазоны	EU 863 – 870 МГц; EU 433 МГц; US 902 – 928 МГц; China 779 – 787 МГц.	

Достоинства технологии LoRaWAN:

– высокая энергоэффективность конечных устройств (до 10 лет);

– большая емкость сети (тысячи устройств на одну базовую станцию);

– отсутствие платы за передачу сообщений;

– использование открытого протокола LoRaWAN;

– простота разворачивания сети;

– надежное шифрование данных;

– поддержка технологии крупными компаниями (IBM, Cisco, Semtech и др.).

Среди перечисленных в начале данного раздела технологий передачи данных технология LoRaWAN наиболее распространена в России.

Примером разворачивания сетей LoRaWAN является проект компании «AURORA Mobile Technologies» по созданию системы поквартирного учета тепла [13]. Результатом внедрения данной системы стала экономия 10% денежных средств управляющей компании [13, с. 4].

Наиболее популярным направлением для применения технологии LoRaWAN являются системы учета электрической энергии. Примеры автоматизированных систем учета электрической энергии представлены на следующих электронных ресурсах [14–18].

Также следует отметить, что в настоящее время в России существуют компании, производящие базовые станции, которые поддерживают протокол обмена LoRaWAN. Примером отечественной базовой станции служит базовая станция «Вега БС 2.2» компании «Вега Абсолют» [19].

Перейдем к рассмотрению сотовой технологии передачи данных NB-IoT.

NB-IoT (Narrowband Internet of Things) – технология беспроводной узкополосной передачи данных, разработанная специально для приложений Интернета вещей [20–24]. Данная технология создана на основе существующих стандартов мобильной связи.

Стандарт NB-IoT был специфицирован соглашением консорциума 3GPP в документе «Release 13» в 2016 году [54].

Основные технические характеристики сетей построенных на технологии NB-IoT представлены в таблице 2.2.

Данные о дальности связи, указанные в таблице 2.2, взяты с электронного ресурса [20].

Основными достоинствами данной технологии являются:

– низкое электропотребление конечных устройств;

– большая емкость сети (тысячи устройств на одну базовую станцию);

– низкая стоимость конечных устройств.

Таблица 2.2 – Технические характеристики сетей NB-IoT

Характеристики	Значение	
Тип используемой модуляции	OFDMA/DSSS	
Ширина полосы, кГц	200	
Скорость передачи данных, кбит/с	Uplink сообщения	Downlink сообщения
	от 1 до 144	от 1 до 200
Дальность, км	до 22	
Тип частотного диапазона	лицензированный	
Используемые частоты, МГц	700, 800, 900	

В конце 2017 года государственная комиссия по радиочастотам (ГКРЧ) разрешила использование ряда полос радиочастот для развертывания сетей, основанных на технологии NB-IoT [25].На данный момент в России только начинается внедрение NB-IoT.

Перейдем к рассмотрению технологий передачи данных Sigfox и Weightless [55–59]. Технические характеристики данных технологий представлены в таблице 2.3.

Таблица 2.3 – Технические характеристики беспроводных технологий передачи Sigfox и Weightless

Характеристика	Sigfox	Weightless
Тип модуляции	BPSK	FDMA / TDMA
Тип частотного диапазона	Нелицензированный (ISM band)	Нелицензированный (ISM band)
Скорость передачи, кбит/с	0,1	от 0,2 до 100

Окончание таблицы 2.3

Характеристика	Sigfox		Weightless
Ширина полосы, кГц	100		12,5
Используемые частоты	868,8; 915		169; 433; 470; 780; 868; 915; 923
Алгоритм шифрования	AES с HMACs		AES-128 / 256
Дальность связи, км	В городских условиях	В открытой местности	В городских условиях
	до 10	до 50	до 2

Технологии передачи данных Sigfox и Wheitghless не распространены в России.

Более детальный обзор технологий беспроводной передачи данных, на основе которых создаются сети LPWAN, представлен в статье [26].

2.1 Выводы по разделу 2

Среди представленных LPWAN технологий беспроводной передачи данных наиболее подходящей для построения автоматизированной системы учета тепловой энергии является технология LoRaWAN. Основными причинами выбора данной технологии служат:

– большой срок работы автономных конечных устройств (до 10 лет);

– простота развертывания сети;

– отсутствие платы за передачу сообщений по сети LoRaWAN;

– большая распространённость технологии на территории России по сравнению с другими энергоэффективными технологиями передачи данных.

К ограничениям данной технологии передачи данных можно отнести:

– малый объем передаваемых данных;

– небольшие скорости передачи данных.

3 ОПИСАНИЕ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ УЧЕТА ТЕПЛОВОЙ ЭНЕРГИИ

3.1 Назначение и цели создания системы

Разрабатываемая система учета тепловой энергии является системой коммерческого учета, поэтому основным назначением системы является обеспечение расчетов между теплосетевыми организациями и потребителями тепловой энергии.

Основными целями создания автоматизированной системы учета являются:

– повышение энергоэффективности текущих систем учета;

– снижение эксплуатационных расходов.

Для достижения поставленных целей необходимо решить перечень задач:

– произвести анализ требований к системам коммерческого учета тепловой энергии;

– произвести анализ существующих решений, требований и аппаратного обеспечения в области учета тепловой энергии;

– произвести анализ технологий беспроводной передачи данных:

1) применяемых для учета тепловой энергии.

2) LPWAN – энергоэффективных сетей дальнего радиуса действия;

– сформировать выводы о применимости и возможных ограничениях использования энергоэффективных сетей дальнего радиуса действия в системах учета тепловой энергии;

– разработать структуру автоматизированной системы учета тепловой энергии на основе энергоэффективных сетей дальнего радиуса действия;

– осуществить выбор технических средств, функционирующих на уровнях сбора, передачи и представления данных;

– разработать программное обеспечение, необходимое для функционирования системы учета;

– произвести проверку работоспособности разработанной системы учета.

Анализ требований, предъявляемых к системам коммерческого учета тепловой энергии, анализ существующий решений и анализ беспроводных технологий передачи данных был произведен в предыдущих разделах работы.

3.2 Функции, выполняемые разрабатываемой системой учета

Автоматизированная система учета тепловой энергии должна выполнять следующие функции:

– сбор показаний с узлов учета;

– сбор, обработка и хранение данных на сервере сбора и хранения данных;

– представление данных в удобном для обслуживающего персонала виде;

– формирование отчетов о количестве потребленной тепловой энергии.

3.3 Структура разрабатываемой системы учета тепловой энергии

На рисунке 3.1 представлена структура разрабатываемой автоматизированной системы учета тепловой энергии на основе энергоэффективных сетей дальнего радиуса действия.

Необходимо отметить, что применение энергоэффективной технологии передачи данных для организации системы учета тепловой энергии наиболее обосновано при условии автономного электропитания элементов узла учета тепловой энергии. Поэтому дальнейшее проектирование системы учета тепловой энергии будет производиться на основе автономных элементов узла учета.

Опишем процесс получения данных о потребленной тепловой энергии в разрабатываемой системе учета.

Тепловычислитель выполняет сбор данных с измерительных преобразователей температуры, расходомеров и производит вычисление потребленной тепловой энергии, и других необходимых параметров.

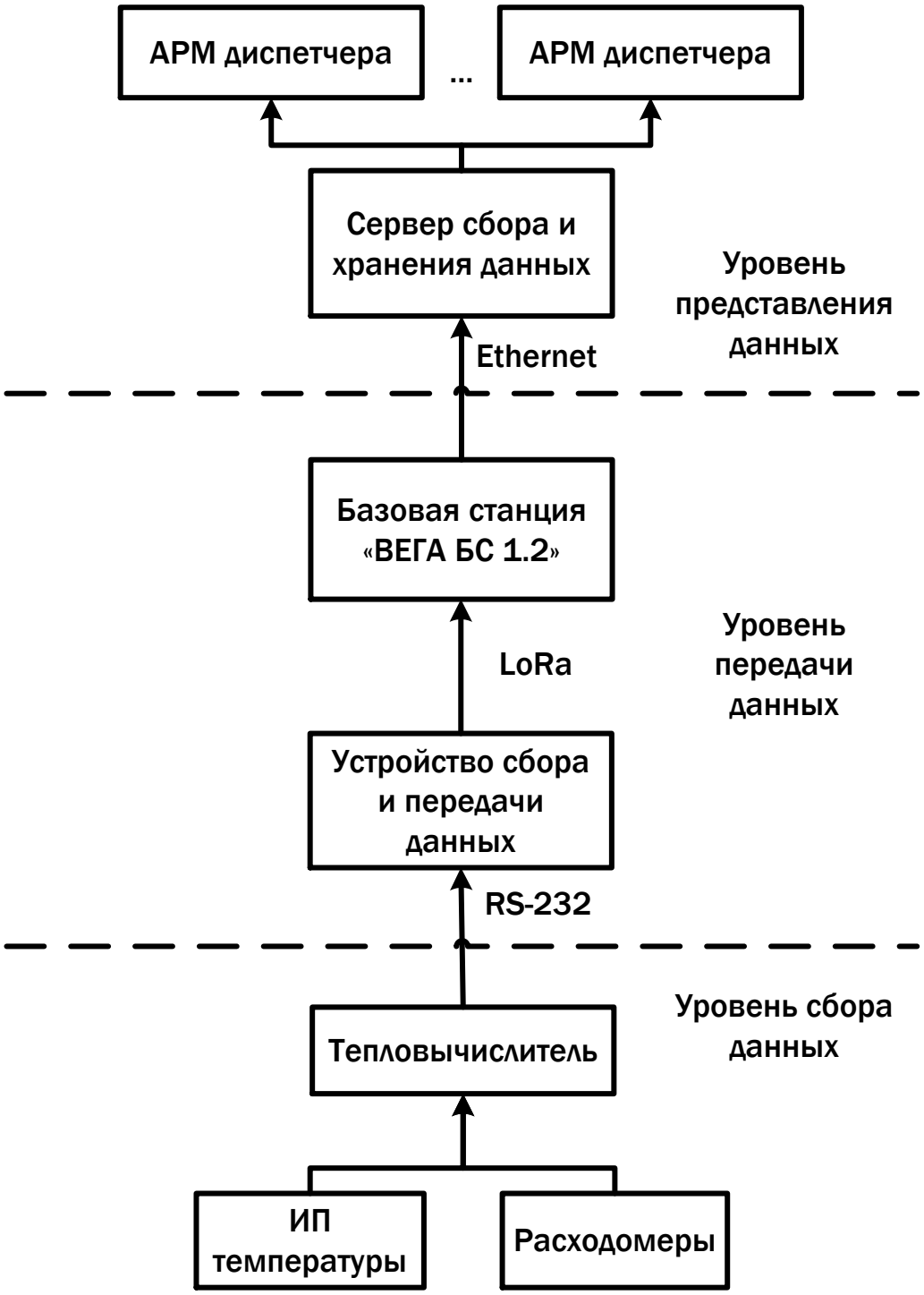

Рисунок 3.1 – Структура разрабатываемой системы учета

По интерфейсу RS-232 к тепловычислителю подключается устройство сбора и передачи данных (УСПД), которое осуществляет периодическое считывание параметров, хранящихся в памяти тепловычислителя. Считанные параметры передаются на основе беспроводной технологии связи LoRa на базовую станцию.

Основной задачей базовой станции является прием данных с УСПД и дальнейшая передача данных на сервер сбора и хранения данных.

С сервера сбора и передачи данных информация о количестве потребленной тепловой энергии и другие параметры поступают на автоматизированное рабочее место диспетчера.

Достоинствами предлагаемой системы учета является отсутствие платы за передачу сообщений от тепловычислителя на уровень представления данных, а также автономное электропитание элементов теплосчетчика. Кроме этого в разрабатываемой системе учета снижены эксплуатационные затраты, по сравнению с системой учета на основе GSM сетей связи.

Необходимо отметить, что в структуре разрабатываемой системы учета в состав теплосчетчика не входит датчик давления, так как для большинства датчиков давления необходим внешний источник питания, а разрабатываемая система учета является автономной.

Согласно правилам коммерческого учета тепловой энергии отсутствие датчика давления допустимо, если тепловая нагрузка систем не превосходит 0,1 Гкал/ч [5]. Поэтому данная система разрабатывается с учетом этого ограничения.

Готовое устройство сбора и передачи данных, осуществляющее преобразование протоколов Modbus RTU (данный протокол должен поддерживать тепловычислитель согласно техническому заданию) в LoRaWAN, на текущий момент времени не было найдено. Поэтому было принято решение самостоятельной разработки устройства сбора и передач данных.

Для осуществления расчетов между теплосетевыми организациями и потребителями тепловой энергии необходимо знать следующие параметры, которые рассчитываются тепловычислителем:

– количество потребленной тепловой энергии;

– время действия нештатных ситуаций;

– время измерения.

На основе вышеизложенного устройству сбора и передачи данных достаточно считывать с тепловычислителя приведенные параметры и передавать данные параметры на базовую станцию.

3.4 Выводы по разделу 3

В данном разделе представлено описание разрабатываемой системы учета тепловой энергии. Приведены цели создания системы учета и функции, которые должна выполнять система.

Ключевым моментом данного раздела является то, что применение энергоэффективных технологий передачи данных наиболее обосновано при наличии автономного электропитания элементов теплосчетчика. Однако применять энергоэффективные технологии передачи данных можно и в узлах учета, в которых имеется подключение к электрической сети, что также имеет ряд преимуществ.

4 ВЫБОР ТЕХНИЧСЕКИХ СРЕДСТВ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УЧЕТА

4.1 Выбор тепловычислителя

Для выбора тепловычислителя сформируем список требований, которым тепловычислитель должен соответствовать.

– иметь стандартные промышленные интерфейсы и протоколы обмена;

– обладать автономным электропитанием;

– иметь возможность расчета тепловой энергии без учета показаний с измерительного преобразователя давления;

На основании приведенных требований произведем выбор автономных тепловычислителей. Перечень автономных тепловычислителей представлен в таблице 4.1.

Таблица 4.1 – Тепловычислители с автономным электропитанием

№	Тепловычислитель	Производитель	Напряжение питания, В	Емкость батареи, А · ч	Тип батареи
1	ВКТ-7-01	НПФ «Теплоком»	3,6	1,9	Литиевая
2	ВКТ-7М-01			7	
3	ТМК-Н20	НПО «Промприбор»		7,2	
4	ТМК-Н30			7,2	
5	ТСРВ-033	ГК «Взлет»		7,5	
6	ТСРВ-043			7,5	
7	КАРАТ-306-1	НПО «Карат»		7,2	
8	КАРАТ-307-4V4T4P			7,2	
9	ЭЛЬФ-01			2,1	

Функциональные возможности автономных тепловычислителей представлены в таблице 4.2.

Таблица 4.2 – Функциональные возможности тепловычислителей

№	Модель	Количество подключаемых датчиков		
		РМ	ТС	ПД
1	ВКТ-7-01	4	2	–
2	ВКТ-7М-01	3	3	3
3	ТМК-Н20	2	2	2
4	ТМК-Н30	3	3	3
5	ТСРВ-033	3	3	–
6	ТСРВ-043	6	5	4
7	КАРАТ-306-1	3	3	3
8	КАРАТ-307-4V4T4P	4	4	4
9	ЭЛЬФ-01	2	2	–

Рассмотрим, какие виды интерфейсов и протоколов поддерживают представленные тепловычислители (см. таблицу 4.3).

Таблица 4.3 – Интерфейсы связи и протоколы обмена тепловычислителей

№	Модель	Интерфейс связи	Протокол обмена
1	ВКТ-7-01	RS-232, RS-485, Ethernet	Modbus RTU
2	ВКТ-7М-01		
3	ТМК-Н20	RS-232, RS-485	
4	ТМК-Н30		
5	ТСРВ-033	RS-232	
6	ТСРВ-043	RS-232, RS-485	
7	КАРАТ-306-1	RS-232, RS-485, M-Bus, радиоинтерфейс, оптический интерфейс	
8	КАРАТ-307-4V4T4P	RS-232, RS-485, M-Bus, радиоинтерфейс, оптический интерфейс, USB	
9	ЭЛЬФ-01	RS-232, RS-485, M-Bus, радиоинтерфейс, оптический интерфейс	

Большинство из представленных тепловычислителей соответствуют указанным требованиям. В качестве тепловычислителя выберем ВКТ-7М-01 компании «Теплоком» [27, 28].

Данный тепловычислитель имеет встроенный интерфейс связи RS-232. Остальные интерфейсы (RS-485, Ethernet) устанавливаются по заказу.

В таблице 4.4 представлены характеристики встроенного интерфейса RS-232.

Таблица 4.4 – Характеристики встроенного интерфейса RS-232

Параметр	Значение
Скорость обмена, Кбит/с	1,2; 2,4; 4,8; 9,6; 19,2
Количество бит данных	8
Количество стоповых бит	2
Контроль честности	нет
Управление потоком	нет

В качестве базовой скорости обмена тепловычислителя по интерфейсу RS-232 выступает скорость 9600 кбит/с. Внешний вид тепловычислителя представлен на рисунке 4.1.

Рисунок 4.1 – Тепловычислитель ВКТ-7М-01

4.2 Выбор измерительного преобразователя температуры

В руководстве по эксплуатации тепловычислителя ВКТ-7М-01 представлены требования к выбору измерительных преобразователей температуры. В таблице 4.5 приведены типы номинальных статических характеристик рекомендуемых для использования с тепловычислителем ВКТ-7М-01.

Подключение измерительных преобразователей температуры к тепловычислителю ВКТ-7М-01 осуществляется по четырехпроводной схеме.

В качестве измерительного преобразователя температуры выберем термометр сопротивления ТЭМ-100 компании «Теплоэнергомонтаж» [29]. Технические характеристики данного термометра сопротивления представлены в таблице 4.6.

Таблица 4.5 – Параметры рекомендуемых для ВКТ-7М-01 термометров сопротивления

Номинальная статическая характеристика	Температурный коэффициент $\alpha, °C^{-1}$
100М	0,00428
100П	0,00391
500П	
Pt100	0,00385
Pt500	

Таблица 4.6 – Технические характеристики ТЭМ-100

Параметр	Значение
Диапазон измеряемых температур	от $-50\ °C$ до $190\ °C$
Номинальная статическая характеристика	Pt100
Температурный коэффициент $\alpha, °C$	0,00385
Максимальный ток измерения, мА	1
Степень защиты от воды и пыли	IP55

Окончание таблицы 4.6

Параметр	Значение
Длина монтажной части, мм	от 50 до 320
Средний срок службы, лет	12

Внешний вид термометра сопротивления ТЭМ-100 представлен на рисунке 4.2.

Рисунок 4.2 – Термометр сопротивления ТЭМ-100

4.3 Выбор расходомера

Перечислим требования, предъявляемые к расходомеру, описанные в руководстве по эксплуатации тепловычислителя ВКТ-7М-01 [28].

В качестве расходомеров применяются только расходомеры, имеющие импульсный выход. Вес импульса должен находиться в диапазоне от 0,0001 до 10000 литров.

Выходная цепь расходомера может быть пассивной (геркон, открытый коллектор) или активной (ТТЛ, КМОП).

Требования, предъявляемые к расходомеру при пассивной выходной цепи расходомера, представлены в таблице 4.7, при активной выходной цепи – в таблице 4.8.

Таблица 4.7 – Требования при пассивной цепи

Параметр	Значение
Частота импульсов, Гц	16
Длительность разомкнутого состояния выходной цепи, мс	более 50
Сопротивление выходной цепи в замкнутом состоянии, кОм	менее 3
Сопротивление выходной цепи в разомкнутом состоянии, МОм	более 3

Таблица 4.8 – Требования при активной цепи

Параметр	Значение
Частота импульсов, Гц	1000
Длительность низкого уровня напряжения, мс	не менее 0,5
Напряжение выходной цепи в состоянии высокого уровня, В	от 2,4 до 5
Напряжение выходной цепи в состоянии низкого уровня, В	$\pm 0{,}4$

На основании приведенных требований произведем выбор расходомера. Учтем, что расходомер должен иметь автономное электропитание.

В качестве расходомера выберем ультразвуковой расходомер РУС-1А научно-производственного объединения «Наука» [30].

Значение максимального и минимального расхода в зависимости от диаметра трубопровода для расходомера РУС-1А приведено в приложении А в таблице А.1.

Данный расходомер имеет пассивный выход типа открытый коллектор. Параметры импульсного выхода расходомера представлены в таблице 4.9.

Таблица 4.9 – Параметры импульсного выхода расходомера

Параметр	Значение
Максимальная частота импульсов, Гц	16
Длительность импульса, мс	1

Окончание таблицы 4.9

Параметр	Значение
Напряжение на открытом транзисторе, В	0,6 – 0,9
Напряжение питания импульсного выхода, В	не более 30
Тип электропитания	от литиевой батареи напряжением 3,6 В
Срок службы батареи	При периоде измерения 4 сек. (рекомендуется для систем отопления) – не менее 4 лет. При периоде измерения 1 сек. (рекомендуется для систем ГВС и ХВС) – не менее 3 лет.

В руководстве по эксплуатации расходомера РУС-1А сказано, что при использовании тепловычислителя ВКТ, импульсный выход расходомера подключается непосредственно к соответствующему входу тепловычислителя без использования внешнего источника питания [30].

Внешний вид ультразвукового расходомера РУС-1А представлен на рисунке 4.3.

Рисунок 4.3 – Ультразвуковой расходомер РУС-1А

4.4 Выбор базовой станции

Основным требованием для выбора базовой станции является поддержка протокола обмена LoRaWAN. В качестве базовой станции выберем базовую станцию «Вега БС 1.2» компании «Вега АБСОЛЮТ» [31].

Внешний вид базовой станции «Вега БС 1.2» представлен на рисунке 4.4.

Рисунок 4.4 – Базовая станция «Вега БС 1.2»

Технические характеристики базовой станции представлены в таблице 4.10.

Таблица 4.10 – Технические характеристики базовой станции

Параметр	Значение
Канал связи с сервером	Ethernet, GSM 3G
Операционная система	Linux
Количество каналов связи	8

Окончание таблицы 4.10

Частотный диапазон	863–870 МГц
Мощность передатчика	до 500 мВт
Дальность радиосвязи в условиях городской застройки	до 5 км
Дальность радиосвязи в условиях сельской местности	до 15 км
Потребляемая мощность, Вт	4
Питание	
Степень защиты	IP67

4.5 Выводы по разделу 4

В данном разделе работы произведен выбор технических средств автоматизированной системы учета тепловой энергии.

Согласно требованиям, предъявляемым к узлу учета, описанным в разделе 3, произведен выбор элементов узла учета с автономным электропитанием. Сформирован список автономных тепловычислителей, которые могут быть включены в состав автоматизированных систем учета тепловой энергии на основе энергоэффективных сетей дальнего радиуса действия. Из данного списка выбран тепловычислитель ВКТ-7М-01, на базе которого осуществлено проектирование автоматизированной системы учета тепловой энергии, описанной в данной работе.

Основным недостатком тепловычислителей, представленных в таблице 4.1, является то, что данные тепловычислители не поддерживают прием показаний с датчиков давления в какой-либо иной форме кроме выходного сигнала 4…20 мА. Для формирования такого сигнала датчикам давления необходим источник питания в большинстве случаев на 12 или 24 В. Следовательно, применение таких датчиков давления в автономной системе учета тепловой энергии либо невозможно, либо для источника питания датчиков давления необходим аккумулятор. Наличие аккумулятора приводит к дополнительным эксплуатационным расходам.

Как уже было сказано в разделе 3, в узле учета тепловой энергии может отсутствовать датчик давления, если тепловая нагрузка в системе теплопотребления не превышает 0,1 Гкал/ч. Поэтому предполагается, что в разрабатываемой системе учета выполняется данное требование и датчик давления для расчета тепловой энергии не нужен.

Исходя из требований, описанных в руководстве по эксплуатации тепловычислителя, выбраны элементы узла учета: измерительный преобразователь температуры ТЭМ-100 и ультразвуковой расходомер с автономным электропитанием РУС-1А.

Для приема сообщений с показаниями тепловычислителя от разрабатываемого устройства сбора и передачи данных выбрана базовая станция «Вега БС 1.2». Основным требованием для выбора базовой станции являлась поддержка протокола LoRaWAN.

5 УСТРОЙСТВО СБОРА И ПЕРЕДАЧИ ДАННЫХ

5.1 Назначение и выполняемые задачи

Устройство сбора и передачи (УСПД) данных предназначено для считывания и последующей передачи на базовую станцию параметров тепловычислителя, таких как: тепловая энергия, время действия нештатных ситуаций, дата и время измерения.

УСПД должно выполнять следующие задачи:

– осуществлять запросы к тепловычислителю и получать ответы по протоколу Modbus RTU.

– передавать данные по протоколу LoRaWAN на базовую станцию;

– управлять модулем приемопередатчика;

– после окончания передачи переходить в режим пониженного электропотребления.

5.2 Состав устройства сбора и передачи данных

Основным элементом устройства сбора и передачи данных является микроконтроллер STM32F103C8T6 [60]. Технические характеристики микроконтроллера представлены в таблице 5.1.

Таблица 5.1 – Основные технические характеристики микроконтроллера STM32F103C8T6

Параметр	Значение
Микропроцессорное ядро	ARM 32-bit Cortex-M3
Максимальная тактовая частота, МГц	72
FLASH-память, Кбит	64

Окончание таблицы 5.1

Параметр	Значение	
Напряжение питания, В	от 2 до 3,6	
Режимы с низким электропотреблением	Sleep, Stop, Standby	
Интерфейсы связи	Название	Количество
	I^2C	2
	SPI	2
	USART	3
	CAN	1
	USB	1
Количество портов ввода/вывода, шт.	37	
Тип корпуса	LQFP48	

Для упрощения реализации устройства сбора и передачи данных возьмем готовую отладочную плату на базе микроконтроллера STM32F103C8T6. Внешний вид отладочной платы представлен на рисунке 5.1.

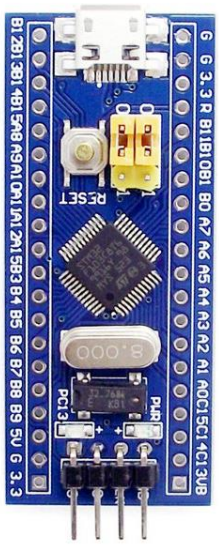

Рисунок 5.1 – Отладочная плата на базе микроконтроллера STM32F103

Схема электрическая принципиальная данной отладочной платы представлена в [61].

Для осуществления обмена по протоколу LoRaWAN между УСПД и базовой станцией выберем модуль приемопередатчика RAK811 [62]. Технические характеристики модуля представлены в таблице 5.2.

Таблица 5.2 – Технические характеристики модуля RAK811

Параметр	Значение		
Поддерживаемые протоколы	LoRaWAN		
Поддерживаемые типы модуляции	LoRa, FSK, GFSK, OOK		
Поддерживаемые полосы частот	RAK811-LF: EU433, CN470; RAK811-HF: EU868, US915, AU915, KR920, AS923, IN865		
Максимальная мощность передатчика, мВт.	100		
Напряжение питания, В	3,3		
Ток потребления	Режим	Значение	Единицы измерения
	Передача	30	мА
	Прием	5,5	мА
	Sleep	7,2	мкА
Интерфейс связи	UART		
Управление приемопередатчиком	при помощи AT-команд		

Параметры обмена данными по UART приемопередатчика представлены в таблице 5.3.

Таблица 5.3 – Параметры обмена по UART модуля RAK811

Параметр	Значение
Скорость обмена, кбит/с	115200
Количество бит данных	8

Окончание таблицы 5.3

Параметр	Значение
Количество стоповых бит	1
Контроль честности	нет
Управление потоком	нет

Внешний вид модуля приемопередатчика RAK811 представлен на рисунке 5.2.

Рисунок 5.2 – Модуль приемопередатчика RAK811

Схема электрическая принципиальная модуля приема передатчика представлена в [63].

Тепловычислитель ВКТ-7М-01 имеет встроенный интерфейс связи RS-232. Поэтому для взаимодействия микроконтроллера и тепловычислителя необходим преобразователь интерфейсов UART-RS-232. В качестве преобразователя интерфейсов выберем трансивер SP3232E [64]. Технические характеристики трансивера представлены в таблице 5.4.

Таблица 5.4 – Технические характеристики трансивера SP3232E

Параметр	Значение
Напряжение питания, В	от 3 до 5,5
Ток потребления при напряжении питания 3,3 В, мА	0,3
Ток потребления в режиме Shutdown при напряжении питания 3,3 В, мкА	1
Максимальное значение входного напряжения на RxIn, В	±15
Выходное значение напряжения на TxOUT, В	±5,4

Для реализации УСПД выберем готовый преобразователь интерфейсов UART-RS-232 на основе микросхемы SP3232E. Внешний вид преобразователя представлен на рисунке 5.3.

Рисунок 5.3 – Преобразователь интерфейсов UART-RS-232 на основе SP3232E

Схема электрическая принципиальная преобразователя интерфейсов представлена в [65].

Схема электрическая принципиальная, схема функциональной структуры устройства сбора и передачи данных представлены в приложении В.

Схема электрическая принципиальная разработана согласно государственным стандартам [32–50].

5.3 Программа для устройства сбора и передачи данных

Тепловычислитель ВКТ-7М-01 производит обмен информацией с внешними устройствами по протоколу Modbus RTU [66]. Из данного протокола тепловычислитель поддерживает две функции: чтение регистров хранения (Read Holding Registers 0x03) и записи нескольких регистров (Write Multiple registers 0x10).

Создадим программу на языке программирования С для микроконтроллера STM32F103, предназначенную для считывания параметров тепловычислителя ВКТ-7М-01.

5.3.1 Описание библиотеки «*modbus_master*»

Разработаем библиотеку, содержащую прототипы функций чтения регистров хранения и записи нескольких регистров.

Программный код библиотеки представлен в листинге Б.1 приложения Б. Кроме прототипов функций, в данном коде содержатся структуры запросов *req_0x03* и *req_0x10*, которые, впоследствии, будут передаваться функциям формирования запросов в виде аргументов.

В первую очередь реализуем функцию расчета контрольной суммы CRC16 согласно алгоритму, описанному в руководстве «MODBUS over Serial Line» [67, с. 39]. Данная функция, впоследствии, будет использоваться в других функциях, формирующих запросы.

Программный код функции расчета контрольной суммы (функция *crc_16*) представлен в листинге Б.2 приложения Б.

Функция *crc_16* принимает в качестве аргументов указатель на массив (*uint8_t *buffer*), в котором содержится запрос, и переменную, содержащую количество элементов в массиве (*uint8_t buffer_size*). Возвращаемым значением функции *crc_16* является контрольная сумма, имеющая тип *uint16_t*.

Реализуем функцию запроса на чтение регистров хранения (Read Holding Registers 0x03) согласно спецификации протокола Modbus RTU [66, с. 15].

Программный код функции чтения регистров хранения (функция *read_hold_reg_0x03*) представлен в листинге Б.2 приложения Б.

Функция *read_hold_reg_0x03* принимает в качестве аргументов: указатель на handle UART'a (*UART_HandleTypeDef *huart*), указатель на структуру запроса (*req_0x03 *request*) и указатель на массив (*uint8_t *txBuffer*), предназначенный для заполнения данными из структуры запроса. Функция *read_hold_reg_0x03* не имеет возвращаемого значения.

Функция *read_hold_reg_0x03* заполняет массив *txBuffer* в соответствии со структурой, представленной в таблице 5.5. После заполнения массива осуществляется передача запроса по UART.

Таблица 5.5 – Структура запроса для функции *read_hold_reg_0x03*

Название элемента запроса	Размер элемента запроса, байт
Адрес опрашиваемого устройства	1
Код функции	1
Начальный адрес	2
Количество регистров	2
Контрольная сумма	2

Реализуем функцию запроса на запись нескольких регистров (Write Multiple Registers 0x10) согласно спецификации протокола Modbus RTU [66, с. 30].

Программный код функции записи нескольких регистров (функция *write_mult_reg_0x10*) представлен в листинге Б.2 приложения Б.

Функция *write_mult_reg_0x10* принимает в качестве аргументов: указатель на handle UART'a (*UART_HandleTypeDef *huart*), указатель на структуру запроса (*req_0x10 *request*) и указатель на массив (*uint8_t *txBuffer*), предназначенный для

заполнения данными из структуры запроса. Функция *write_mult_reg_0x10* не имеет возвращаемого значения.

Функция *write_mult_reg_0x10* заполняет массив *txBuffer* в соответствии со структурой, представленной в таблице 5.6. После заполнения массива осуществляется передача запроса по UART.

Таблица 5.6 – Структура запроса для функции *write_mult_reg_0x10*

Название элемента запроса	Размер элемента запроса, байт
Адрес опрашиваемого устройства	1
Код функции	1
Начальный адрес	2
Количество регистров	2
Количество записываемых байт данных	1
Байты данных	$N \cdot 2$
Контрольная сумма	2

Обозначения, представленные в таблице 5.6: N – количество регистров.

5.3.2 Описание библиотеки «*vkt_7m_01*»

Для получения архивных данных с тепловычислителя ВКТ-7М-01 микроконтроллер должен выполнить ряд запросов. Порядок следования запросов описан в справочном руководстве тепловычислителя «Протокол обмена» [51].

Согласно справочному руководству сформируем очередность запросов, которые должен выполнить микроконтроллер, имеет вид:

1) запрос «Начало сеанса связи»;

2) запрос «Чтение текущей даты и времени»;

3) запрос на запись типа значений;

4) запрос на запись перечня элементов для чтения;

5) запрос на запись даты;

6) запрос на чтение данных.

После выполнения представленных запросов тепловычислитель отправит требуемые архивные данные, начиная с указанной даты.

При получении запроса тепловычислитель формирует ответ. Ответом может являться:

– блок данных, в соответствии с запросом;

– подтверждение запроса, связанное с записью в тепловычислитель необходимых параметров;

– исключительный ответ, содержащий код возникшего при выполнении запроса исключения.

Разработаем библиотеку, содержащую функции, реализующие приведенные запросы и выполняющие разбор ответов тепловычислителя. В таблице 5.7 представлено соответствие запросов и функций библиотеки, осуществляющих запросы и разборы ответов.

Таблица 5.7 – Соответствие названия запросов и функций библиотеки

Название запроса	Название функции, реализующей запрос	Название функции, реализующей разбор ответа на запрос
Запрос «Начало сеанса связи»	*start_session()*	*parse_resp_0x10()*
Запрос на запись типа значений	*write_type_of_values()*	
Запрос на запись перечня элементов для чтения	*write_reading_list()*	
Запрос на запись даты	*write_date()*	
Запрос «Чтение текущей даты и времени»	*read_current_date()*	*parse_current_date()*
Запрос на чтение данных	*read_data()*	*parse_resp_data()*

Прототипы функций, представленных в таблице 5.7, описаны в заголовочном файле «*vkt_7m_01.h*». Код заголовочного файла представлен в листинге Б.3 приложения Б.

Также в заголовочном файле разрабатываемой библиотеки присутствует описание структуры *date* и перечисления *state*. Структура *date* предназначена для хранения даты и времени. Перечисление *state* содержит различные возвращаемые функциями разбора ответов состояния.

Рассмотрим, какие аргументы передаются представленным в таблице 5.7 функциям, а также типы возвращаемых функциями значений.

Все функции, реализующие запросы имеют в качестве аргументов: handle UART'а (*UART_HandleTypeDef *huart*), необходимый для последующей передачи запроса; указатель на массив (*uint8_t *txBuffer*), который в каждой функции запроса передается в функции *write_mult_reg_0x10* или *read_hold_reg_0x03*; адрес тепловычислителя (*uint8_t address*), для которого предназначен запрос. Типом возвращаемого функциями запросов значения является *void*.

Функции запроса имеют одинаковую структуру. В первую очередь в каждой функции по UART передаются два байта 0xFF. Передача данных байтов необходима для вывода тепловычислителя из «спящего» режима [51, с. 9].

Далее объявляется и заполняется структура *req_0x03* или *req_0x10* в зависимости от типа запроса. После заполнения структуры необходимые параметры передаются в функции *read_hold_reg_0x03()* или *write_mult_reg_0x10()*.

Поясним назначение отдельных функций запросов. Функция *write_type_of_values()* предназначена для записи типа возвращаемых тепловычислителем данных в ответ на запрос «Чтение данных».

Функция *write_type_of_values()* кроме перечисленных выше аргументов принимает тип записываемого значения (*uint8_t type*). В качестве типа записываемого значения выступают следующие типы [51, с. 16]:

– 0 – часовой архив;

– 1 – суточный архив;

– 2 – месячный архив;

– 3 – итоговый архив;

– 4 – текущие значения;

– 5 – итоговые текущие значения;

– 6 – свойства тепловычислителя.

Функция *write_reading_list()* предназначена для формирования запроса с перечнем элементов, значения которых вернет тепловычислитель, в случае запроса «Чтение данных».

Для проверки параметров, присутствующих в каждом ответе тепловычислителя на запрос, создана функция *check_response()*.

Реализация всех функций запросов и разбора представлена в листинге Б.4 приложения Б.

5.4 Описание программы main.c

Код основной программы устройства сбора и передачи данных представлен в листинге Б.5 приложения Б.

Программа выполняет последовательные запросы согласно порядку указанному в пункте 5.3.2. В случае если какой-либо из запросов не привел к ожидаемому результату (ошибка CRC, наличие исключения), то данный запрос повторяется определенное количество раз.

Если после одинаковых запросов все же не удалось считать требуемые данные, то на базовую станцию передается сообщение об ошибке.

При успешном считывании показаний тепловычислителя на базовую станцию передаются считанные данные.

После считывания параметров тепловычислителя происходит установление сеанса связи с базовой станцией. Если сеанс связи установлен, то на базовую станцию передаются либо считанные данные, либо сообщение об ошибке, в зависимости от результата предыдущего обмена с тепловычислителем.

В случае успешной передачи считанных параметров тепловычислителя на базовую станцию, происходит настройка alarm'а часов реального времени на сутки.

Если сеанс связи с базовой станцией не установлен или считывание параметров тепловычислителя не произведено, то alarm часов реального времени настраивается на час.

После настройки alarm'а происходит переход в режим пониженного электропотребления Standby. Выход из режима Standby осуществляется по срабатыванию alarm'а RTC. После чего описанная последовательность действий повторяется.

Опишем процедуру получения произвольного объема данных по UART микроконтроллера STM32.

Так как стандартные функции приема библиотеки HAL позволяют принимать только строго определенное количество символов, было принято решение разработать собственную реализацию функции приема. Результатом реализации служит функция *receive()*.

При вызове данной функции запускается прием одного байта кадра. Одновременно с запуском приема запускается таймер. Прерывания таймера настроены на определенный период, по истечении которого считается, что прием сообщения закончен.

В случае если после запуска приема по UART'у пришел байт данных, то вызывается прерывание по UART. Принятый байт данных записывается в буфер приема, в переменную *countOfBytes* прибавляется единица. После чего происходит сброс счетчика таймера и запускается прием очередного байта.

Если после последнего приема байта данных прошло время, равное периоду прерывания таймера, то вызывается прерывание таймера, в котором отменяется запуск приема очередного байта. После этого прием данных считается завершенным и из буфера, содержащего принятые байты, можно осуществлять чтение данных.

Расчет настроек таймера представлен в пункте 5.5.

Схема работы программы представлена в приложении В.

5.5 Необходимые расчеты для УСПД

Произведем расчет коэффициента предделителя таймера при заданных параметрах тактовой частоты работы микроконтроллера, верхнего предела AutoReload регистра, желаемой частоты прерываний таймера.

Прерывания таймера микроконтроллера настраиваются на определенный период. Через данное время после приема последнего байта по UART'у должно срабатывать прерывание таймера и останавливать последующий прием байтов.

Частота прерываний таймера определяется по формуле (5.1

$$f = \frac{f_{clk}}{(k+1)(N_{\text{вп}}+1)}, \tag{5.1}$$

где f_{clk} – тактовая частота работы микроконтроллера;

k – коэффициент предделителя таймера;

$N_{\text{вп}}$ – верхний предел AutoReload регистра.

Значение тактовой частоты микроконтроллера равно: $f_{clk} = 8$ МГц. Верхний предел AutoReload регистра выберем равным: $N_{\text{вп}} = 999$.

Согласно протоколу обмена тепловычислителя ВКТ-7М-01 после приема кадра тепловычислитель выжидает интервал тишины 62,5 мс и после это формирует ответ. Следовательно, прерывания таймера должны происходить не менее чем через 62,5 мс после приема. Выберем период прерывания таймера равным 200 мс.

Расчет частоты прерываний таймера также производится по формуле (5.2:

$$f = \frac{1}{T}, \tag{5.2}$$

где T – период прерываний таймера.

Приравняем левые части выражений (5.1) и (5.2) и выразим коэффициент предделителя k, получим выражение (5.3).

$$k = \frac{T f_{clk}}{N_{\text{вп}} + 1} - 1. \quad (5.3)$$

Подставим в формулу (5.3) приведенные выше значения параметров, получим значение коэффициента предделителя:

$$k = \frac{0{,}2 \cdot 8000000}{999 + 1} - 1 = 1599. \quad (5.4)$$

5.6 Конфигурация микроконтроллера STM32F103 в STM32CubeMX

Произведем конфигурацию микроконтроллера при помощи программы STM32CubeMX [68, 69]. В качестве среды разработки выберем TrueSTUDIO [70].

Настроим выходы UART'а, осуществляющего обмен данных с преобразователем интерфейсов UART-RS-232, к которому подключается тепловычислитель. Параметры обмена тепловычислителя были представлены в таблице 4.4. В качестве скорости обмена данными выберем скорость 9600 кбит/с. Результат настройки UART2 представлен на рисунке 5.4.

Перейдем к настройке UART'а, который производит обмен данными с модулем приемопередатчика RAK811. Параметры обмена приёмопередатчика представлены в таблице 5.3. Настроим UART3 в соответствии с данными параметрами обмена. Результат настройки UART3 представлен на рисунке 5.5.

Во вкладке настройки прерываний (NVIC Settings) для каждого из UART'ов активируем глобальные прерывания (USART global interrupt).

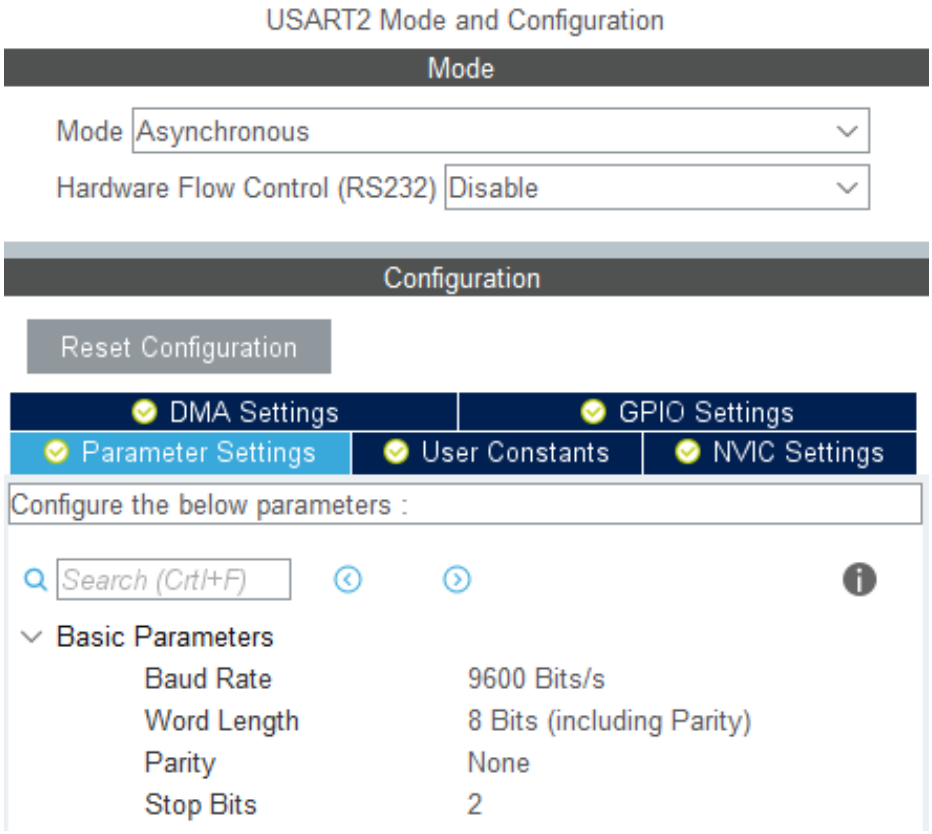

Рисунок 5.4 – Настройка UART2

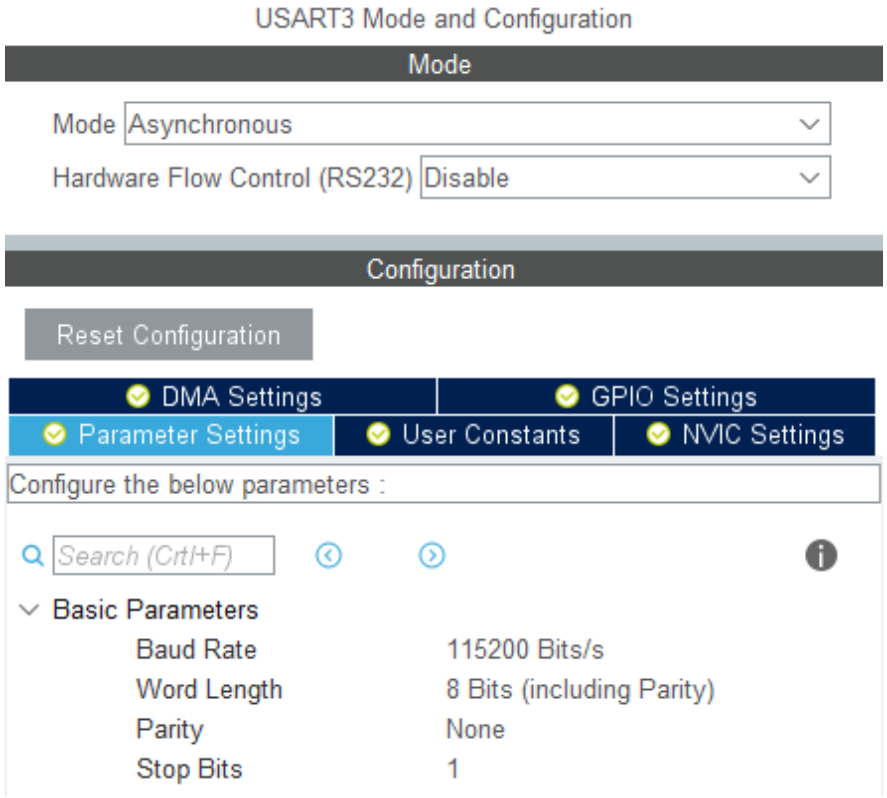

Рисунок 5.5 – Настройка UART3

Произведем настройку часов реального времени. Для этого активируем тактирование RTC. В качестве настроек даты и времени оставим стандартные настройки. Установим время, через которое сработает alarm, равным 30 секундам. Это необходимо для того, чтобы alarm не срабатывал сразу после включения микроконтроллера. Тридцати секунд после включения достаточно, чтобы микроконтроллер успел установить необходимое время, через которое должен произойти очередной выход из режима пониженного электропотребления.

Результат настройки часов реального времени представлен на рисунке 5.6.

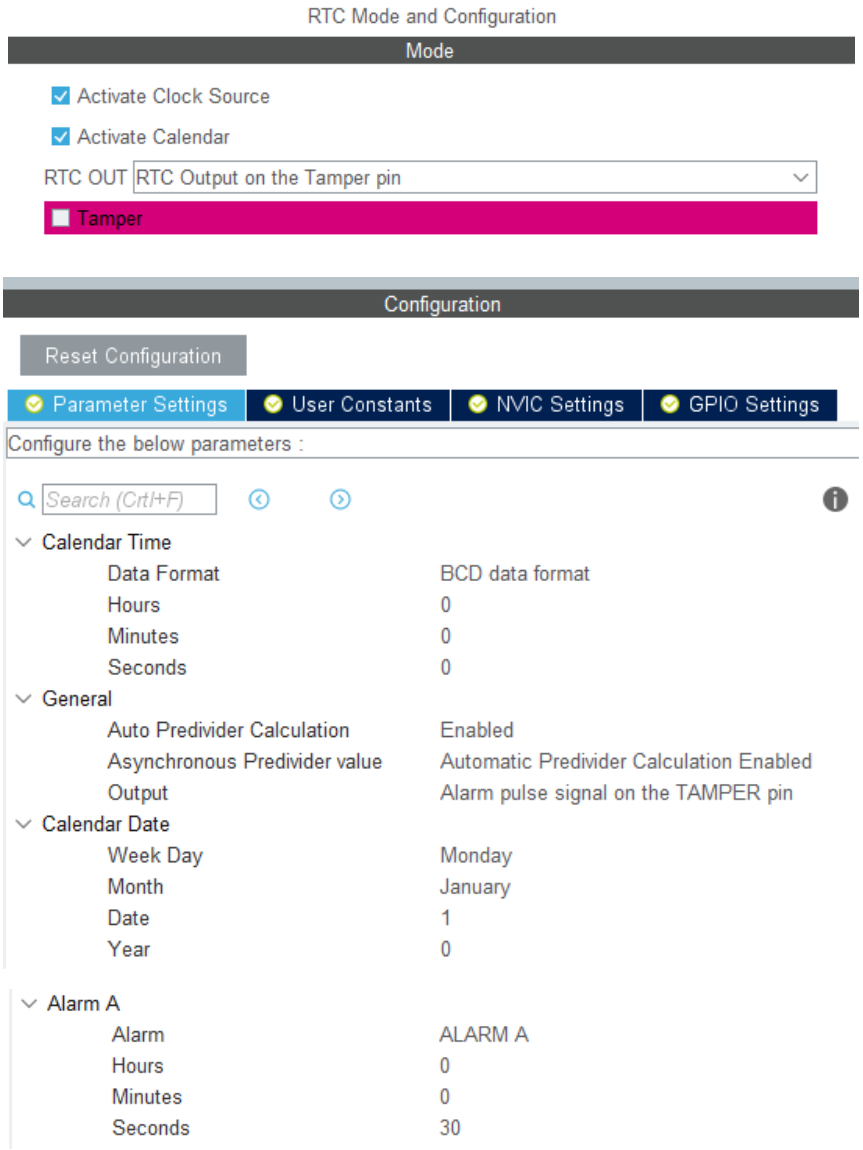

Рисунок 5.6 – Настройка RTC 2

Настроим таймер, который необходим при процедуре приема данных по UART. Расчет параметров таймера представлен в пункте 5.5. Установим параметры таймера в соответствии с данными расчетами. Результат настройки представлен на рисунке 5.7.

Для включения / выключения светодиодов, необходимых для оповещения оператора о состоянии установления связи сконфигурируем пины PB0, PB1 в качестве выходов. Результат конфигурации PB0, PB1 представлен на рисунке 5.8.

Конфигурация всех выходов микроконтроллера приведена на рисунке 5.9. Настройка тактирования микроконтроллера отображена на рисунке 5.10.

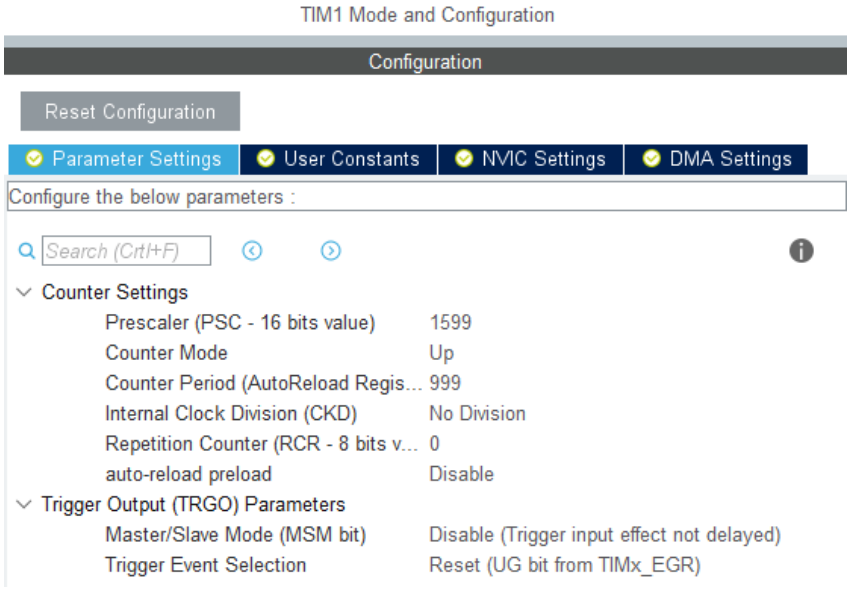

Рисунок 5.7 – Настройка таймера

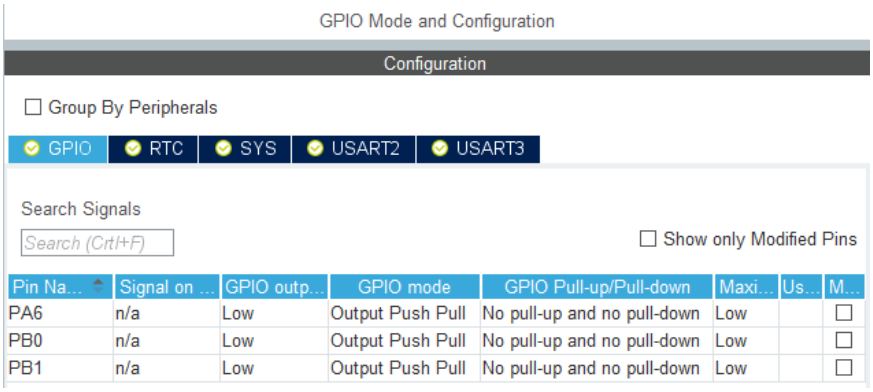

Рисунок 5.8 – Конфигурация выходов PB0, PB1

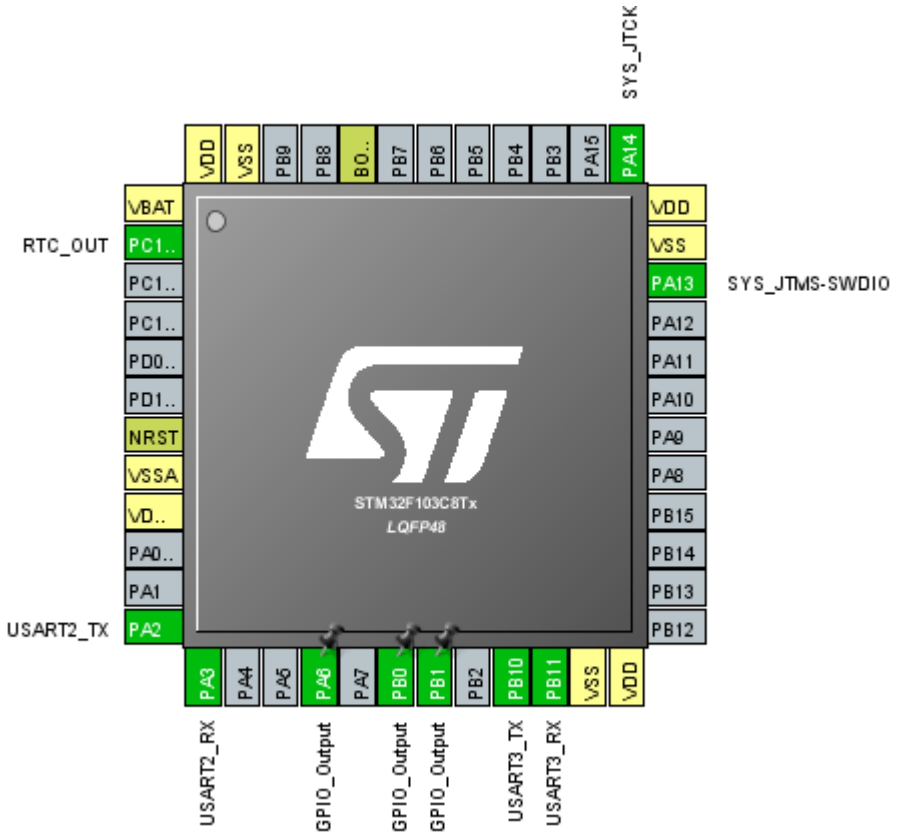

Рисунок 5.9 – Конфигурация пинов микроконтроллера

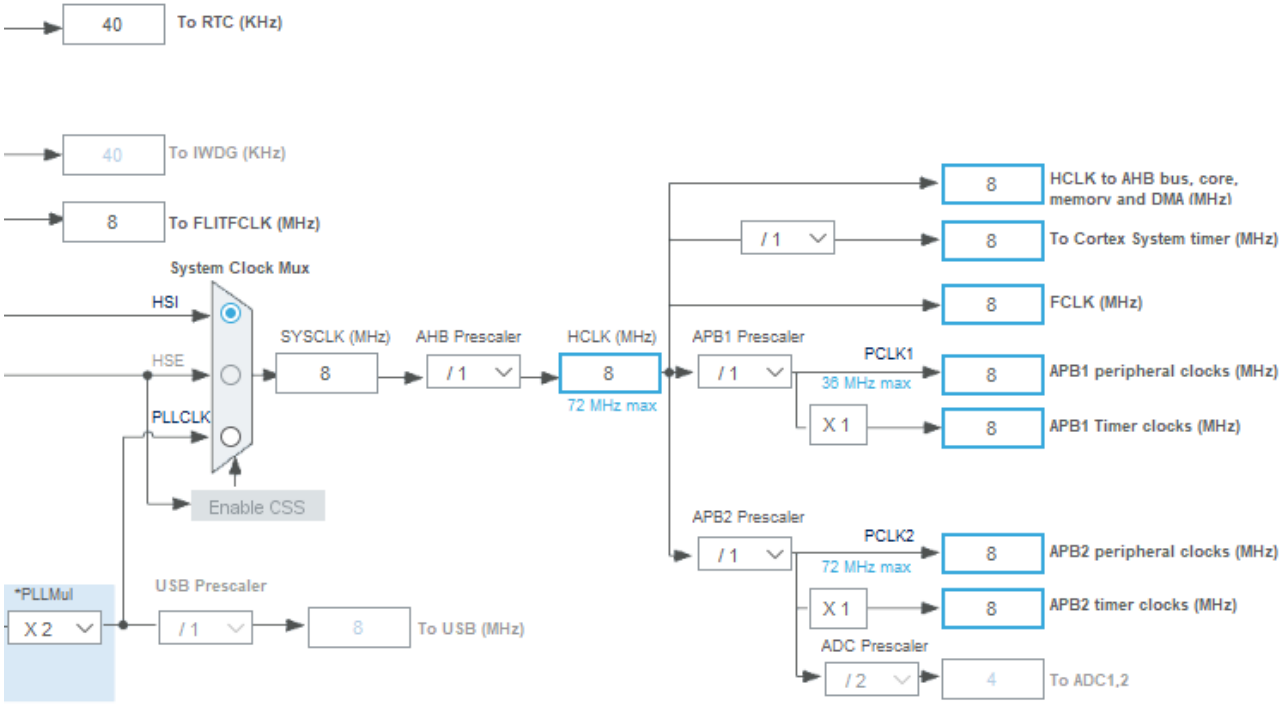

Рисунок 5.10 – Настройка тактирования микроконтроллера

5.7 Прототип УСПД

Внутреннее содержимое прототипа устройства сбора и передачи данных представлено на рисунке 5.11.

Из рисунка 5.11 видим, что основу устройства сбора и передачи данных составляют выбранные в пункте 5.2 элементы: отладочная плата STM32F103, преобразователь интерфейсов UART–RS-232, модуль приемопередатчика RAK811.

Кроме элементов, описанных в пункте 5.2, в состав устройства сбора и передачи входят:

– батарейный отсек под аккумулятор типоразмера 18650;

– аккумуляторная батарея Li-pol 5200 мА · ч.

– переходник UFL гнездо – SMA штекер;

– разъем GX12M-4B;

– кнопка с фиксацией для включения и выключения питания;

– корпуса для светодиодов;

– светодиоды диаметром 3 мм;

– макетная плата 80x120;

– корпус для радиоэлектронной аппаратуры.

Все элементы УСПД размещены на макетной плате. Пайка компонентов УСПД осуществлялась на основе схемы электрической принципиальной, представленной в приложении.

Лицевая часть устройства сбора и передачи данных представлена на рисунке 5.12.

На лицевую часть УСПД выведены два светодиода, предназначенных для оповещения оператора об установлении / не установлении соединения с тепловычислителем и базовой станцией.

Также на лицевой части расположена кнопка включения / выключения питания устройства сбора и передачи данных.

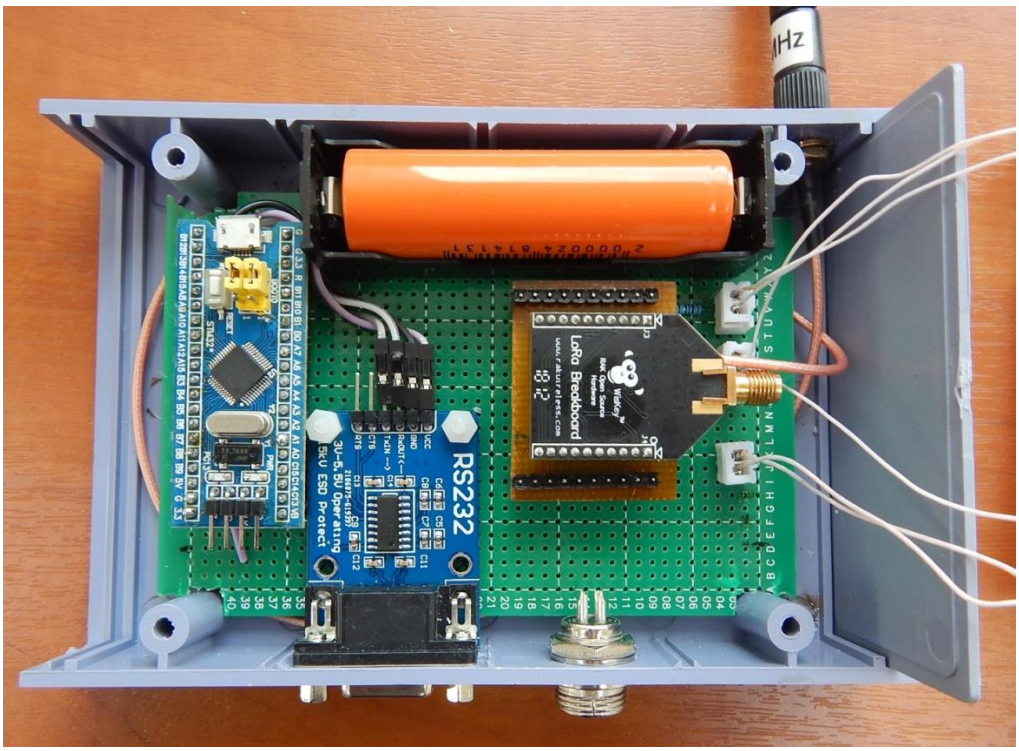

Рисунок 5.11 – Внутреннее содержимое прототипа УСПД

Рисунок 5.12 – Лицевая часть УСПД

5.8 Выводы по разделу 5

В рамках данного раздела описаны основные этапы разработки устройства сбора и передачи данных. Базовыми компонентами УСПД являются: отладочная плата на базе микроконтроллера STM32F103C8T6, модуль приемопередатчика RAK811, а также преобразователь интерфейсов UART-RS-232.

Для УСПД были разработаны библиотеки, содержащие функции запросов, необходимых для считывания параметров тепловычислителя, а также функции разбора ответов.

На основе разработанных библиотек написана основная программа для устройства сбора и передачи данных.

В конце раздела представлен прототип УСПД, выполняющий считывание параметров тепловычислителя и передачу данных параметров на базовую станцию.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы выполнены следующие задачи в рамках разработки автоматизированной системы учета тепловой энергии на основе энергоэффективных сетей дальнего радиуса действия:

– произведен анализ существующих решений в области учета тепловой энергии;

– выявлены основные преимущества и недостатки существующих систем учета, в которых передача данных осуществляется на основе сетей сотовой связи;

– выполнен анализ энергоэффективных технологий беспроводной передачи данных, применение которых позволит снизить текущие затраты на эксплуатацию систем учета на основе GSM;

– разработана структура системы автоматизированного учета тепловой энергии на основе энергоэффективной технологии беспроводной передачи данных LoRa;

– произведен выбор технических средств с автономным электропитанием для узла учета тепловой энергии;

– выполнен выбор базовой станции для приема данных с тепловычислителей;

– разработано устройство сбора и передачи данных, осуществляющее считывание параметров с тепловычислителя о потребленной тепловой энергии и передачу данных параметров на базовую станцию;

– разработана основная программа и требуемые библиотеки на языке программирования С, необходимые для функционирования устройства сбора и передачи данных.

Достоинствами предлагаемой системы учета являются:

– отсутствие платы за передачу данных от тепловычислителя на верхний уровень;

– автономное питание элементов узла учета и устройства сбора и передачи данных.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Теплоэнергетика и централизованное теплоснабжение России 2015 – 2016 годы. – https://minenergo.gov.ru/system/download-pdf/10850/80685.
2. Федеральный закон о теплоснабжении. – http://www.consultant.ru/document/cons_doc_LAW_102975/.
3. Теплоснабжение: Учебное пособие / В. Е. Козин, Т. А. Левина, А. П. Марков и др. – Т34 М.: Высш. школа, 1980. – 408 с. ил.
4. Об утверждении методики осуществления коммерческого учета тепловой энергии, теплоносителя. – http://docs.cntd.ru/document/499086231.
5. О коммерческом учете тепловой энергии, теплоносителя. – http://docs.cntd.ru/document/499058683.
6. GSM диспетчеризация приборов учета. – http://www.mnppsaturn.ru/?topic_id=116.
7. Контроллер БКД-ПК-RF. – https://www.mnppsaturn.ru/ftp/public/doc/bkd-pk-rf/re%20BKD-PK-RF-DIN%20103.pdf.
8. LPWAN. – https://iot.ru/theme/lpwan.
9. LPWAN – большой обзор сетей дальнего радиуса для интернета вещей. – https://voltiq.ru/the-guide-to-low-power-wide-area-networks/.
10. Архитектура LoRaWAN сетей. – http://lo-ra.ru/lorawan-networks/.
11. Технология LoRa. – http://lo-ra.ru/.
12. Выдрин Д. Ф., Ситдиков Д. Р. Основные параметры беспроводной технологии LoRaWAN // Academy. 2019. №2 (41). – https://cyberleninka.ru/article/n/osnovnye-parametry-besprovodnoy-tehnologii-lorawan.
13. LoRaWAN над Санкт-Петербургом. – https://controleng.ru/wp-content/uploads/4958.pdf.
14. Система учета на базе технологии LoRaWAN. – http://www.ankomplus.ru/lorawan.shtml.

15. Система автоматизированного учета электроэнергии Комета LoRaWAN. – http://tancos.ru/cometa_lorawan.html.

16. АСКУЭ для СНТ. – https://waviot.ru/reshenia/poselki/askue-dlya-snt/.

17. АСКУЭ Zenner-Minol LoRaWAN. – http://minol-service.ru/katalog/askue-minol-zenner/68-askue-zenner-minol-lorawan.

18. Меркурий LoRaWAN решение для АСКУЭ. – http://77cs.ru/ascue-lorawan.

19. Вега БС-2.2 - базовая станция. – http://iotvega.com/product/bs02-2.

20. Технология NB-IoT. – https://iot.ru/wiki/nb-iot.

21. Технология узкополосного интернета вещей (NB-IoT) в сети мобильной связи. – http://1234g.ru/novosti/uzkopolosnyj-internet-veshchej-nb-iot.

22. Технология связи NB-IoT. – https://itechinfo.ru/content/nb-iot.

23. NB-IoT – интернет вещей в сим-карте. – https://stoneforest.ru/look/gadgets/nb-iot/.

24. Модули NB-IoT. – http://ultran.ru/catalog/nb-iot.

25. Заседание ГКРЧ от 28 декабря 2017 года (протокол №17-44). – https://digital.gov.ru/ru/documents/5875/.

26. Кумаритова, Д. Л. Обзор и сравнительный анализ технологий LPWAN сетей/ Д.Л. Кумаритова, Р.В. Киричек // Информационные технологии и телекоммуникации. – 2016. – № 4. – С. 33–48.

27. Тепловычислитель ВКТ-7М. – http://teplocom-sale.ru/catalogue/?SECTION_ID=146&ELEMENT_ID=351263.

28. ВКТ-7М-01. Вычислитель количества теплоты. Руководство по эксплуатации. – http://teplocom-sale.ru/upload/iblock/826/ВКТ-7М-01%20РЭ%20версия%204.1%20август%202018.pdf.

29. Термометры сопротивления ТЭМ 100. – http://www.logika-consortium.ru/komplektnye-postavki/product/tem-100/.

30. Ультразвуковой расходомер РУС-1А. Руководство по эксплуатации. – http://rus1r.ru/документация-лицензии-и-сертификаты/item/rus-1a-rukovodstvo-po-ekspluatatsii.

31. Базовая станция Вега БС-1.2. – http://iotvega.com/product/bs01-2.

32. ГОСТ 2.001-93. Единая система конструкторской документации. Общие положения. – М.: Изд-во стандартов, 2001. – 8 с.

33. ГОСТ 2.102-68. Единая система конструкторской документации. Виды и комплектность конструкторских документов. – М.: Стандартинформ, 2007. – 8 с.

34. ГОСТ 2.104-2006. Единая система конструкторской документации. Основные надписи. – М.: Стандартинформ, 2006. – 20 с.

35. ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам. – М.: Стандартинформ, 2005. – 30 с.

36. ГОСТ 2.106-96. Единая система конструкторской документации. Текстовые документы. – М.: Стандартинформ, 2007. – 32 с.

37. ГОСТ 2.301-68. Единая система конструкторской документации. Форматы. – М.: Изд-во стандартов, 2000. – 3 с.

38. ГОСТ 2.303-68. Единая система конструкторской документации. Линии. – М.: Стандартинформ, 2007. – 8 с.

39. ГОСТ 2.304-68. Единая система конструкторской документации. Шрифты чертёжные. – М.: Изд-во стандартов, 1979. – 20 с.

40. ГОСТ 2.701-2008. ЕСКД. Схемы. Виды и типы. Общие требования к выполнению. – М.: Изд-во стандартов, 2008. – 27 с.

41. ГОСТ 2.702-75. ЕСКД. Правила выполнения электрических схем. – М.: Изд-во стандартов, 1975. – 45 с.

42. ГОСТ 2.708-81. Единая система конструкторской документации. Правила выполнения электрических схем цифровой вычислительной техники. – М.: Стандартинформ, 2005. – 15 с.

43. ГОСТ 2.709-89. ЕСКД. Правило выполнения электрических схем цифровой вычислительной техники. – М.: Изд-во стандартов, 1989. – 34 с.

44. ГОСТ 2.710-81. ЕСКД. Обозначения буквенно-цифровые в электрических схемах. – М.: Изд-во стандартов, 1981. – 15 с.

45. ГОСТ 2.721-74. ЕСКД. Обозначения общего применения. – М.: Изд-во стандартов, 1974. – 23 с.

46. ГОСТ 2.723-68. ЕСКД. Катушки индуктивности, дроссели, трансформаторы, автотрансформаторы, магнитные усилители. – М.: Изд-во стандартов, 1968. – 48 с.

47. ГОСТ 2.728-74. ЕСКД. Резисторы. Конденсаторы. – М.: Изд-во стандартов, 1974. – 38 с.

48. ГОСТ 2.730-73. ЕСКД. Полупроводниковые приборы. – М.: Изд-во стандартов, 1973. – 12 с.

49. ГОСТ 2.743-91. ЕСКД. Элементы цифровой техники. – М.: Изд-во стандартов, 1991. – 22 с.

50. ГОСТ 2.755-87. ЕСКД. Устройства коммутационные и контактные соединения. – М.: Изд-во стандартов, 1987. – 53 с.

51. Реализация протокола Modbus для связи с тепловычислителями ВКТ-7М. – http://teplocom-sale.ru/catalogue/documentation.php?SECTION_ID=112.

52. What is the LoRaWAN Specification?. – https://lora-alliance.org/about-lorawan.

53. LoRaWAN 1.1 Specification. – https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf.

54. 3GPP Standards for the Internet-of-Things. – ftp://www.3gpp.org/Information/presentations/presentations_2016/2016_11_3gpp_Standards_for_IoT.pdf.

55. Sigfox Technology Overview. – https://www.sigfox.com/en/sigfox-iot-technology-overview.

56. Weightless Specification. – http://www.weightless.org/about/weightless-specification.

57. NB-IoT, LoRa, Sigfox. – https://www.link-labs.com/blog/nb-iot-vs-lora-vs-sigfox.

58. Sigfox overview. – http://www.emcu.eu/sigfox-vs-lora/.

59. Sigfox vs. LoRa. – https://engineering.eckovation.com/sigfox-vs-lora-one-prefer-iot-device/.

60. Datasheet STM32F103x8. – https://www.st.com/resource/en/datasheet/stm32f103c8.pdf.

61. STM32F103 schematic board. – https://wiki.stm32duino.com/images/c/c1/Vcc-gnd.com-STM32F103C8-schematic.pdf.

62. Datasheet RAK811 Lora Module. – http://docs.rakwireless.com/en/LoRa/RAK811/Hardware_Specification/RAK811_LoRa_Module_Datasheet_V1.4.pdf.

63. Low Power Long Range Transceiver. – http://docs.rakwireless.com/en/LoRa/RAK811/Hardware_Specification/RAK811_HF_Schematics.pdf.

64. Datasheet SP3222E/3232E. – https://www.sparkfun.com/datasheets/Components/General/sp3222_3232e.pdf.

65. RS-232 board. – https://static.chipdip.ru/lib/160/DOC001160766.pdf.

66. Modicon Modbus Protocol Reference Guide v1.1b3. – http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf.

67. MODBUS over Serial Line. Specification and Implementation Guide V1.02. – http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf.

68. STM32Cube initialization code generator. – https://www.st.com/en/development-tools/stm32cubemx.html.

69. STM32Cube. – https://www.st.com/en/ecosystems/stm32cube.html

70. TrueSTUDIO. – https://atollic.com/truestudio/.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

Таблица А.1 – Ограничения по расходу для расходомера РУС-1А в зависимости от диаметра трубопровода

Условный диаметр, Ду, мм	Расход, $м^3/ч$		
	Максимальный расход, $Q_{МАКС}$	Переходный расход, $Q_П$	Минимальный расход, $Q_{МИН}$
15	5	0,07	0,03
25	10	0,17	0,07
32	34	0,7	0,3
40	54	1,5	0,5
50	85	2,5	0,6
65	144	3,2	1,6
80	218	4	1,0
100	340	6	1,25
150	765	9	1,7
200	1360	12	2,5
250	2120	15	3,0
300	3100	18	3,5
400	5400	25	5
500	8500	30	6
600	12200	35	7,5
700	16700	40	8,5
800	21800	50	10

ПРИЛОЖЕНИЕ Б

Листинги библиотек *modbus_master, vkt_7m_01* и основной программы УСПД

Листинг Б.1 – Header file библиотеки «*modbus_master*»

```
#ifndef MODBUS_MASTER_H_
#define MODBUS_MASTER_H_

#ifndef __STM32F1xx_HAL_H
#include "stm32f1xx_hal.h"
#endif

#include <stdint.h>

typedef struct
{
     uint8_t  address;
     uint16_t startingAddress;
     uint16_t quantityOfHoldReg;
} req_0x03;

typedef struct
{
     uint8_t  address;
     uint16_t startingAddress;
     uint16_t quantityOfRegisters;
     uint8_t  byteCount;
     uint8_t  registerValue[246];
} req_0x10;

void read_hold_reg_0x03(UART_HandleTypeDef *huart,
          req_0x03 *request,
          uint8_t *txBuffer);

void write_mult_reg_0x10(UART_HandleTypeDef *huart,
          req_0x10 *request,
          uint8_t *txBuffer);

uint16_t crc_16(uint8_t *buffer, uint8_t buffer_size);

#endif /* MODBUS_MASTER_H_ */
```

Листинг Б.2 – Source file библиотеки «*modbus_master*»

```
#include "modbus_master.h"

void read_hold_reg_0x03(UART_HandleTypeDef *huart,
          req_0x03 *request,
          uint8_t *txBuffer)
{
     txBuffer[0] = request->address;
     txBuffer[1] = 0x03;
     txBuffer[2] = (uint8_t)(request->startingAddress >> 8);
     txBuffer[3] = (uint8_t)(request->startingAddress & 0x00FF);
     txBuffer[4] = (uint8_t)(request->quantityOfHoldReg >> 8);
     txBuffer[5] = (uint8_t)(request->quantityOfHoldReg & 0x00FF);

     uint16_t sizeOfBuffer = 8;
     uint16_t crc = crc_16(txBuffer, sizeOfBuffer-2);

     txBuffer[6] = (uint8_t)(crc >> 8);
     txBuffer[7] = (uint8_t)(crc & 0x00FF);


     HAL_UART_Transmit(huart, txBuffer, sizeOfBuffer, 0xFFFF);
}

void write_mult_reg_0x10(UART_HandleTypeDef *huart,
          req_0x10 *request,
          uint8_t *txBuffer)
{
     txBuffer[0] = request->address;
     txBuffer[1] = 0x10;
     txBuffer[2] = (uint8_t)(request->startingAddress >> 8);
     txBuffer[3] = (uint8_t)(request->startingAddress & 0x00FF);
     txBuffer[4] = (uint8_t)(request->quantityOfRegisters >> 8);
     txBuffer[5] = (uint8_t)(request->quantityOfRegisters & 0x00FF);
     txBuffer[6] = request->byteCount;

     for(uint8_t i = 0; i < request->byteCount; i++)
     {
          txBuffer[i + 7] = request->registerValue[i];
     }

     uint16_t sizeOfBuffer = request->byteCount + 9;
     uint16_t crc = crc_16(txBuffer, sizeOfBuffer-2);

     txBuffer[request->byteCount + 7] = (uint8_t)(crc >> 8);
     txBuffer[request->byteCount + 8] = (uint8_t)(crc & 0x00FF);

     HAL_UART_Transmit(huart, txBuffer, sizeOfBuffer, 0xFFFF);
}
```

Окончание листинга Б.2

```
uint16_t crc_16(uint8_t *buffer, uint8_t buffer_size)
{
     uint8_t temp = 0;
     uint16_t crc = 0xFFFF;

     for(uint8_t byte = 0; byte < buffer_size; byte++)
     {
          crc = crc ^ buffer[byte];

          for(uint8_t j = 0; j < 8; j++)
          {
               temp = crc & 0x0001;
               crc = crc >> 1;

               if(temp)
               {
                    crc = crc ^ 0xA001;
               }
          }
     }

     temp = crc & 0x00FF;
     crc = (crc >> 8) | (temp << 8);

     return crc;
}
```

Листинг Б.3 – Header file библиотеки «*vkt_7m_01*»

```
#ifndef VKT_7M_01_H_
#define VKT_7M_01_H_


#include "modbus_master.h"


typedef struct
{
     uint8_t second;
     uint8_t minute;
     uint8_t hour;
     uint8_t day;
     uint8_t month;
     uint8_t year;
} date;
```

Продолжение листинга Б.3

```
typedef enum
{
     OK,
     EXCEPTION,
     EMPTY,
     CRC_ERROR
} state;

// Функции запросов

void start_session(UART_HandleTypeDef *huart,
          uint8_t *txBuffer,
          uint8_t address);

void read_current_date(UART_HandleTypeDef *huart,
          uint8_t *txBuffer,
          uint8_t address);

void write_type_of_values(UART_HandleTypeDef *huart,
          uint8_t *txBuffer,
          uint8_t address,
          uint8_t type);

void write_reading_list(UART_HandleTypeDef *huart,
          uint8_t *txBuffer,
          uint8_t address);

void write_date(UART_HandleTypeDef *huart,
          uint8_t *txBuffer,
          date *date,
          uint8_t address);

void read_data(UART_HandleTypeDef *huart,
          uint8_t *txBuffer,
          uint8_t address);


// Функции обработки ответов

state parse_resp_0x10(uint8_t *response, uint8_t respSize);

state parse_current_date(uint8_t *response, date *currentDate,
          uint8_t respSize);

state parse_resp_data(uint8_t *response, uint8_t respSize,
          uint8_t *data);
```

Окончание листинга Б.3

```
state check_response(uint8_t *response, uint8_t respSize);

#endif /* VKT_7M_01_H_ */
```

Листинг Б.4 – Source file библиотеки «*vkt_7m_01*»

```
#include "vkt_7m_01.h"

uint8_t dontSleep[2] = {0xFF, 0xFF};

void start_session(UART_HandleTypeDef *huart,
          uint8_t *txBuffer,
          uint8_t address)
{
     HAL_UART_Transmit(huart, dontSleep, 2, 0xFFFF);

     txBuffer[0] = address;
     txBuffer[1] = 0x10;
     txBuffer[2] = 0x3F;
     txBuffer[3] = 0xFF;
     txBuffer[4] = 0x00;
     txBuffer[5] = 0x00;
     txBuffer[6] = 0xCC;
     txBuffer[7] = 0x80;
     txBuffer[8] = 0x00;
     txBuffer[9] = 0x00;
     txBuffer[10] = 0x00;

     uint16_t sizeOfBuffer = 13;

     uint16_t crc = crc_16(txBuffer, sizeOfBuffer-2);
     txBuffer[11] = (uint8_t)(crc >> 8);
     txBuffer[12] = (uint8_t)(crc & 0x00FF);

     HAL_UART_Transmit(huart, txBuffer, sizeOfBuffer, 0xFFFF);
}


void read_current_date(UART_HandleTypeDef *huart,
          uint8_t *txBuffer,
          uint8_t address)
{
     req_0x03 request;

     HAL_UART_Transmit(huart, dontSleep, 2, 0xFFFF);

     request.address = address;
```

Продолжение листинга Б.4

```
    request.startingAddress = 0x3FFB;
    request.quantityOfHoldReg = 0x0000;

    read_hold_reg_0x03(huart, &request, txBuffer);
}


void write_type_of_values(UART_HandleTypeDef *huart,
        uint8_t *txBuffer,
        uint8_t address,
        uint8_t type)
{
    req_0x10 request;

    HAL_UART_Transmit(huart, dontSleep, 2, 0xFFFF);

    request.address = address;
    request.startingAddress = 0x3FFD;
    request.quantityOfRegisters = 0x0000;
    request.byteCount = 0x02;
    request.registerValue[0] = type;
    request.registerValue[1] = 0x00;

    write_mult_reg_0x10(huart, &request, txBuffer);
}


void write_reading_list(UART_HandleTypeDef *huart,
        uint8_t *txBuffer,
        uint8_t address)
{
    HAL_UART_Transmit(huart, dontSleep, 2, 0xFFFF);

    req_0x10 request;

    request.address = address;
    request.startingAddress = 0x3FFF;
    request.quantityOfRegisters = 0x0000;
    request.byteCount = 0x06;
    request.registerValue[0] = 0x0C;
    request.registerValue[1] = 0x00;
    request.registerValue[2] = 0x00;
    request.registerValue[3] = 0x40;
    request.registerValue[4] = 0x04;
    request.registerValue[5] = 0x00;

    write_mult_reg_0x10(huart, &request, txBuffer);
}
```

Продолжение листинга Б.4

```
void write_date(UART_HandleTypeDef *huart,
          uint8_t *txBuffer,
          date *date,
          uint8_t address)
{
     req_0x10 request;

     HAL_UART_Transmit(huart, dontSleep, 2, 0xFFFF);

     request.address = address;
     request.startingAddress = 0x3FFB;
     request.quantityOfRegisters = 0x0000;
     request.byteCount = 0x04;
     request.registerValue[0] = date->day;
     request.registerValue[1] = date->month;
     request.registerValue[2] = date->year;
     request.registerValue[3] = date->hour;

     write_mult_reg_0x10(huart, &request, txBuffer);
}

void read_data(UART_HandleTypeDef *huart, uint8_t *txBuffer,
          uint8_t address)
{
     req_0x03 request;

     HAL_UART_Transmit(huart, dontSleep, 2, 0xFFFF);

     request.address = address;
     request.startingAddress = 0x3FFE;
     request.quantityOfHoldReg = 0x0000;

     read_hold_reg_0x03(huart, &request, txBuffer);
}

state parse_resp_0x10(uint8_t *response, uint8_t respSize)
{
     state stateOfResp = check_response(response, respSize);
     return stateOfResp;
}

state parse_current_date(uint8_t *response, date *currentDate,
          uint8_t respSize)
{
     state stateOfResp = check_response(response, respSize);

     if(stateOfResp != OK)
     {
```

Продолжение листинга Б.4

```
          return stateOfResp;
     }

     currentDate->day = response[3];
     currentDate->month = response[4];
     currentDate->year = response[5];
     currentDate->hour = response[6];
     currentDate->minute = response[7];
     currentDate->second = response[8];

     return OK;
}

state parse_resp_data(uint8_t *response, uint8_t respSize,
          uint8_t *data)
{
     state stateOfResp = check_response(response, respSize);

     if(stateOfResp != OK)
     {
          return stateOfResp;
     }

     data[0] = response[3];
     data[1] = response[4];
     data[2] = response[5];
     data[3] = response[6];

     return stateOfResp;
}


state check_response(uint8_t *response, uint8_t respSize)
{
     // Проверка на наличие ответа
     if(respSize == 0)
     {
          return EMPTY;
     }

     // Проверка контрольной суммы
     uint16_t recvCRC;
     uint8_t crcLow;
     uint8_t crcHigh;

     crcLow = response[respSize-2];
     crcHigh = response[respSize-1];
```

Окончание листинга Б.4

```
    uint16_t checksum = crc_16(response, respSize-2);
    recvCRC = (crcLow << 8) | crcHigh;

    if(checksum != recvCRC)
    {
        return CRC_ERROR;
    }

    // Проверка на наличие исключений
    if((response[1] & 0x80) != 0)
    {
        return EXCEPTION;
    }

    return OK;
}
```

Листинг Б.5 – Source file main.c

```
#include "main.h"
#include "modbus_master.h"
#include "vkt_7m_01.h"

#define COUNT_OF_REPEATS 3U

#define PORT_1 GPIOB
#define PIN_1_1 GPIO_PIN_0
#define PIN_1_2 GPIO_PIN_1

typedef enum
{
    START_SESSION,
    READ_DATE,
    WRITE_TYPE_OF_VALUES,
    WRITE_READING_LIST,
    WRITE_DATE,
    READ_DATA,
} request;

RTC_HandleTypeDef hrtc;
TIM_HandleTypeDef htim1;

UART_HandleTypeDef huart2;
UART_HandleTypeDef huart3;

// Служебные буферы приема и передачи
uint8_t txBuffer[256] = {0};
uint8_t rxBuffer[1] = {0};
```

Продолжение листинга Б.5

```
// Буфер, содержащий принятые данные
uint8_t buffer[256] = {0};

// Переменная, содержащая количество
// принятых байт данных
uint8_t countOfBytes = 0;

uint8_t respSize = 0;
volatile uint8_t isReadyToRecv = 1;

state parseState;
request next_request;
request previous_request;

date currentDate;

uint8_t data[30] = {0};

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_USART3_UART_Init(void);
static void MX_TIM1_Init(void);
static void MX_RTC_Init(void);

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
	if(huart == &huart2)
	{
		if(isReadyToRecv != 1)
		{
			htim1.Instance->CNT = 0;
			buffer[countOfBytes] = rxBuffer[0];
			countOfBytes++;
			HAL_UART_Receive_IT(&huart2, rxBuffer, 1);
		}
	}
}


void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
	if(htim == &htim1)
	{
		HAL_TIM_Base_Stop_IT(&htim1);
		HAL_UART_AbortReceive_IT(&huart2);
		isReadyToRecv = 1;
		respSize = countOfBytes;
```

Продолжение листинга Б.5

```
            countOfBytes = 0;
      }
}

void receive()
{
      HAL_UART_Receive_IT(&huart2, rxBuffer, 1);
      htim1.Instance->SR &= ~0x0001;
      htim1.Instance->CNT = 0;
      HAL_TIM_Base_Start_IT(&htim1);
      isReadyToRecv = 0;

      while(isReadyToRecv != 1);
}

int main(void)
{
  HAL_Init();
  SystemClock_Config();

  MX_GPIO_Init();
  MX_USART2_UART_Init();
  MX_USART3_UART_Init();
  MX_TIM1_Init();
  MX_RTC_Init();

  RTC_AlarmTypeDef alarm;
  RTC_TimeTypeDef time;
  RTC_TimeTypeDef time1;

  uint8_t message[14] = "at+join=otaa\r\n";

  while (1)
  {
       next_request = START_SESSION;

       for(uint8_t i = 0; i < COUNT_OF_REPEATS; )
       {
            previous_request = next_request;

            switch(next_request)
            {
            case START_SESSION:
                 start_session(&huart2, txBuffer, 1);
                 receive();
                 parseState = parse_resp_0x10(buffer, respSize);
                 next_request = READ_DATE;
                 break;
```

Продолжение листинга Б.5

```
            case READ_DATE:
                read_current_date(&huart2, txBuffer, 1);
                receive();
                parseState = parse_current_date(buffer,
                          &currentDate, respSize);
                next_request = WRITE_TYPE_OF_VALUES;
                break;
            case WRITE_TYPE_OF_VALUES:
                write_type_of_values(&huart2, txBuffer, 1, 1);
                receive();
                parseState = parse_resp_0x10(buffer, respSize);
                next_request = WRITE_READING_LIST;
                break;
            case WRITE_READING_LIST:
                write_reading_list(&huart2, txBuffer, 1);
                receive();
                parseState = parse_resp_0x10(buffer, respSize);
                next_request = WRITE_DATE;
            case WRITE_DATE:
                currentDate.day = currentDate.day - 1;
                currentDate.hour = 23;
                write_date(&huart2, txBuffer, &currentDate, 1);
                receive();
                parseState = parse_resp_0x10(buffer, respSize);
                next_request = READ_DATA;
                break;
            case READ_DATA:
                read_data(&huart2, txBuffer, 1);
                receive();
                parseState = parse_resp_data(buffer,
                          respSize, data);

                next_request = START_SESSION;
                break;
            }

            if(parseState == OK)
            {
                i = 0;
                if(next_request == START_SESSION) {break;}
            }
            else
            {
                ++i;
                next_request = previous_request;
            }
            HAL_Delay(50);
        }
```

Продолжение листинга Б.5

```
if(parseState == OK)
{
     HAL_GPIO_WritePin(PORT_1, PIN_1_1, GPIO_PIN_SET);
     HAL_Delay(1000);
     HAL_GPIO_WritePin(PORT_1, PIN_1_1, GPIO_PIN_RESET);
}
else
{
     HAL_GPIO_WritePin(PORT_1, PIN_1_2, GPIO_PIN_SET);
     HAL_Delay(1000);
     HAL_GPIO_WritePin(PORT_1, PIN_1_2, GPIO_PIN_RESET);
}

HAL_Delay(500);

time1.Hours = 0;
time1.Minutes = 0;
time1.Seconds = 0;

HAL_UART_Transmit(&huart3, message, 14, 0xFFFF);
receive();

if((buffer[0] == 'o') && (buffer[1] == 'k'))
{
     HAL_GPIO_WritePin(PORT_1, PIN_1_1, GPIO_PIN_SET);
     HAL_Delay(1000);
     HAL_GPIO_WritePin(PORT_1, PIN_1_1, GPIO_PIN_RESET);

     data[4] = currentDate.day;
     data[5] = currentDate.month;
     data[6] = currentDate.year;
     data[7] = currentDate.minute;
     data[8] = currentDate.hour;
     HAL_UART_Transmit(&huart3, data, 9, 0xFFFF);

     time.Hours = 23;
     time.Minutes = 59;
     time.Seconds = 59;
     alarm.AlarmTime = time;
     alarm.Alarm = RTC_ALARM_A;
}
else
{
     HAL_GPIO_WritePin(PORT_1, PIN_1_2, GPIO_PIN_SET);
     HAL_Delay(1000);
     HAL_GPIO_WritePin(PORT_1, PIN_1_2, GPIO_PIN_RESET);

     data[0] = 'e';
```

Продолжение листинга Б.5

```
            data[1] = 'r';
            data[2] = 'r';
            data[3] = 'o';
            data[4] = 'r';

            time.Hours = 1;
            time.Minutes = 0;
            time.Seconds = 0;
            alarm.AlarmTime = time;
            alarm.Alarm = RTC_ALARM_A;
            HAL_UART_Transmit(&huart3, data, 5, 0xFFFF);
        }

        HAL_RTC_SetTime(&hrtc, &time1, RTC_FORMAT_BCD);

        HAL_RTC_SetAlarm(&hrtc, &alarm, RTC_FORMAT_BCD);

        __HAL_PWR_CLEAR_FLAG(PWR_FLAG_WU);
        HAL_PWR_EnterSTANDBYMode();
    }
}

void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};

  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI|
  RCC_OSCILLATORTYPE_LSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
  RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.LSIState = RCC_LSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|
  RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_PCLK1|
  RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
```

Продолжение листинга Б.5

```
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) !=
  HAL_OK)
  {
    Error_Handler();
  }
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_RTC;
  PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_LSI;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}

static void MX_RTC_Init(void)
{
  RTC_TimeTypeDef sTime = {0};
  RTC_DateTypeDef DateToUpdate = {0};
  RTC_AlarmTypeDef sAlarm = {0};

  hrtc.Instance = RTC;
  hrtc.Init.AsynchPrediv = RTC_AUTO_1_SECOND;
  hrtc.Init.OutPut = RTC_OUTPUTSOURCE_ALARM;
  if (HAL_RTC_Init(&hrtc) != HAL_OK)
  {
    Error_Handler();
  }

  sTime.Hours = 0x0;
  sTime.Minutes = 0x0;
  sTime.Seconds = 0x0;

  if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BCD) != HAL_OK)
  {
    Error_Handler();
  }
  DateToUpdate.WeekDay = RTC_WEEKDAY_MONDAY;
  DateToUpdate.Month = RTC_MONTH_JANUARY;
  DateToUpdate.Date = 0x1;
  DateToUpdate.Year = 0x0;

  if (HAL_RTC_SetDate(&hrtc, &DateToUpdate, RTC_FORMAT_BCD) !=
  HAL_OK)
  {
    Error_Handler();
  }
  sAlarm.AlarmTime.Hours = 0x0;
  sAlarm.AlarmTime.Minutes = 0x0;
  sAlarm.AlarmTime.Seconds = 0x30;
```

Продолжение листинга Б.5

```
  sAlarm.Alarm = RTC_ALARM_A;
  if (HAL_RTC_SetAlarm_IT(&hrtc, &sAlarm, RTC_FORMAT_BCD) != HAL_OK)
  {
    Error_Handler();
  }
}

static void MX_TIM1_Init(void)
{

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 1599;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 999;
  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.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig)
  != HAL_OK)
  {
    Error_Handler();
  }
}

static void MX_USART2_UART_Init(void)
{
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_2;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
```

Окончание листинга Б.5

```
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
}

static void MX_USART3_UART_Init(void)
{
  huart3.Instance = USART3;
  huart3.Init.BaudRate = 115200;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart3) != HAL_OK)
  {
    Error_Handler();
  }
}

static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0|GPIO_PIN_1, GPIO_PIN_RESET);

  /*Configure GPIO pins : PB0 PB1 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1;
  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);

}

void Error_Handler(void)
{
  // Служебная функция, генерируемая библиотекой HAL
}
```

ПРИЛОЖЕНИЕ В

СХЕМЫ И ЧЕРТЕЖИ