МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт машиностроения
(наименование института полностью)

Кафедра «Промышленная электроника»
(наименование)

11.03.04 Электроника и наноэлектроника
(код и наименование направления подготовки/специальности)

Электроника и робототехника
(направленность (профиль)/специальности)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Система удаленной телеметрии и управления загородным жилым объектом

Обучающийся М.В. Шавеко
(Инициалы Фамилия) (личная подпись)

Руководитель канд. техн. наук, доцент М.В. Позднов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант О.А. Головач
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

Название дипломной работы: «Система удаленной телеметрии и управления загородным жилым объектом».

Выпускная работа состоит из введения, 5 глав, заключение, 7 таблиц, 2 приложения, 43 рисунка, списка литературы, включая зарубежные источники, и графической части на 7 листах формата А1.

Ключевым вопросом разработки является разработка и внедрение надежной и доступной системы удаленной телеметрии и контроля загородного жилого объекта. Это предполагает интеграцию технологий Arduino, GSM и датчиков температуры, реле для включения аварийных обогревателей, а также создание программного модуля для обработки данных и взаимодействия с удаленным сервером.

Цель работы - дать комплексный обзор разработки и внедрения системы удаленной телеметрии и управления для загородного жилого комплекса. Это включает в себя выбор и интеграцию соответствующих аппаратных компонентов, разработку программного модуля для обработки данных и взаимодействия с удаленным сервером, а также применение технологий GSM и датчиков температуры.

Дипломная работа разделена на следующие части: постановка целей и задач, анализ аналогичных устройств на рынке выбор элементов схемы; разработка электрической схемы и печатной платы, разработка кода, проведение ряда экспериментов.

В конце исследования представлена работа, которой является практическая реализация системы удаленной телеметрии и управления для загородного жилого комплекса с возможностью контроля температуры с помощью SMS-сообщений и интерфейса удаленного сервера.

Подводя итоги, мы бы хотели подчеркнуть, что данная работа представляет собой значительный шаг вперед в области систем удаленной телеметрии и контроля для загородной жилой недвижимости.

ABSTRACT

The title of the graduation work is “System for Remote Telemetry and Control of a Suburban Residential Property”.

The senior paper consists of an introduction, 5 parts, a conclusion, 7 tables, 2 applications, 43 figures, list of references including foreign sources and the graphic part on 7 A1 sheets.

The key issue of the thesis is the development and implementation of a reliable and accessible system for remote telemetry and control of a suburban residential property. This involves the integration of Arduino, GSM and temperature sensors, relays, as well as creating a program module for data processing and interaction with a remote server.

The aim of the work is to give a comprehensive overview of the design and implementation of a remote telemetry and control system for a suburban residential property. This includes the selection and integration of appropriate hardware components, the development of a software module for data processing and interaction with a remote server, and the application of GSM and temperature sensor technologies.

The graduation work may be divided into several logically connected parts which are: setting goals and objectives, analyzing similar devices on the market selection of circuit elements; development of the electrical circuit and printed circuit board, code development, conducting a number of experiments.

Finally, we present the work, the result of which is the practical implementation of a remote telemetry and control system for a suburban residential property with the possibility of temperature control using SMS messages and a remote server interface.

In conclusion, we’d like to stress that this work represents a significant step forward in the field of remote telemetry and control systems for suburban residential properties.

Содержание

Введение

В настоящее время развитие технологий умного дома приводит к повышению комфорта, безопасности и экономичности жизни в доме. Одним из направлений является разработка систем, обеспечивающих контроль и регулирование температуры в доме, особенно в зимний период, когда отключение газа может привести к серьезным проблемам в системе отопления.

Смарт-системы для отопления дома являются одним из наиболее популярных и востребованных устройств для умного дома и системы энергосбережения. Они позволяют удаленно контролировать и регулировать температуру в доме через SMS- сообщения, а также получать оповещения о падении температуры до критического уровня в случае отключения газа.

Эти системы обеспечивают комфорт, безопасность, а также помогает экономить на расходах на отопление. Кроме того, есть возможность в систему интегрировать разные датчики обширного спектра функций, таких как датчики света, которые автоматически регулируют освещение в доме.

Популярность смарт-систем растет по причине легкости и простоты использования. Кроме того, все больше распространяется «Интернет вещи» (internet of things, IoT), где через облако специализированных сервисов можно управлять различными электронными устройствами. Необходим только выход в интернет [11].

1 Постановка задачи

Главная цель этой выпускной работы - разработать умную систему для мониторинга и управления отоплением и электроприборами в загородном доме. Для этого используются знания в области микроэлектроники, схемотехники, а также программирования микроконтроллеров и мобильных устройств. Итогом работы станет создание программно-аппаратной системы, с помощью которой пользователь сможет взаимодействовать с домом через SMS и GSM-модуль. Эта система будет работать автономно, контролируя отопление и электроприборы, а также сообщать пользователю о текущем состоянии и возможных проблемах.

Кроме того, умная система сможет отправлять данные о температуре в доме через интернет на специальный веб-сервер. Пользователь сможет просматривать графики и статистику температур за разные периоды, что поможет анализировать эффективность системы отопления и оптимизировать её работу. В случае, если температура в доме опустится ниже критического уровня, система автоматически включит аварийный источник тепла.

1.1 Поиск и анализ технических параметров аналогичных устройств

При создании системы мониторинга температуры для умного дома необходимо сравнить её с уже существующими на рынке решениями, чтобы выявить их сильные и слабые стороны в сравнении с нашим устройством.

Системы мониторинга температуры, такие как RTU5023, ALONIO T2 и Полюс GSM Термо, предлагают разнообразные функции и возможности для обеспечения комфорта и безопасности в доме. Они позволяют пользователям удаленно контролировать температуру и получать уведомления о критических изменениях, что особенно важно в холодное время года или при авариях с отоплением.

Каждое из этих устройств обладает своими уникальными преимуществами и недостатками, которые нужно учитывать при выборе подходящего варианта для конкретного дома или проекта. Важно оценить не только технические характеристики, но и удобство использования, стоимость и возможность интеграции с существующими системами умного дома.

Рассмотрим их подробнее:

— RTU5023 - это устройство, которое поддерживает различные сети связи, включая GSM, GPRS, 3G и 4G, а также облачные технологии. Оно позволяет мониторить не только температуру, но и влажность, аналоговые сигналы, напряжение и состояние питания. RTU5023 поддерживает протоколы Modbus RTU Over TCP и Modbus TCP для интеграции с облачными платформами. Однако, использование GSM модуля SIM800L ограничивает его пропускную способность, так как он работает в стандарте 2G, и устройство не имеет графического интерфейса для отображения параметров [23]. Устройство приведено на рисунке 1.1.

Рисунок 1.1 – RTU5023

— ALONIO T2 - это устройство, которое контролирует температуру и наличие электричества, оповещая пользователя через SMS-сообщения. Оно

просто в установке и эксплуатации, и может отправлять SMS на пять номеров мобильных телефонов. Недостатком является необходимость использования обычной 2G SIM-карты и ограниченный функционал по сравнению с более сложными системами [16]. Устройство показано на рисунке 1.2.

Рисунок 1.2 - ALONIO T2

— Полюс GSM Термо - это устройство, предназначенное для мониторинга температуры окружающей среды и оповещения владельца через сеть GSM. Оно способно отправлять тревожные извещения, но информация о его технических параметрах и функциональных возможностях ограничена, и может потребоваться дополнительное оборудование для полноценного мониторинга [17]. Устройство представлено на рисунке 1.3.

Рисунок 1.3 - Полюс GSM Термо

Для более наглядного представления и сравнения этих систем и нашего устройства сведем данные в таблицу 1.

Таблица 1 – Сравнение аналогов системы с разрабатываемой

Название	Цена, Р	Тип связи	Тип оповещения	Количество номеров	Автономный режим	Графический интерфейс
RTU5023	7790	GSM	SMS	10	Есть, 8 часов	Нет
ALONIO T2	5500	GSM	SMS + звонок на телефон	5	Нет	Нет
Полюс GSM Термо	5700	GSM	SMS + Звонок на телефон	5	Есть, 1 год	Нет
Разрабатываемая система	5000	GSM	SMS + звонок на телефон	50+	Есть, 30 часов	Есть

1.2 Критерии для разработки устройства

Критерии для разработки устройства системы удаленной телеметрии и управления загородным жилым объектом таковы:

1) Стабильная передача данных: Гарантированная передача данных о состоянии объекта (температура, состояние систем отопления/охлаждения и т.д.) без потерь и задержек даже при неблагоприятных условиях сигнала.

2) Энергоэффективность: Минимизация энергопотребления устройства для обеспечения длительного автономного функционирования при использовании батарей или других источников питания.

3)Удобство использования: Удобство установки и интуитивно понятная настройка устройства для конечного пользователя.

4) Интуитивный интерфейс: Легкость использования интерфейса управления для обеспечения комфорта и понятности для пользователей.

5) Защита данных: Обеспечение защиты данных, передаваемых и хранимых в системе, от несанкционированного доступа.

6) Надежная передача данных: Выбор оптимальной технологии передачи данных для обеспечения надежной и стабильной связи с удаленным сервером и управления системой.

7) Климатическая устойчивость: Обеспечение нормальной работы устройства при широком диапазоне температур и влажности.

8) Стабильность работы: Гарантия стабильной работы устройства независимо от внешних условий и воздействий окружающей среды.

9) Автономность: Устройство должно быть способно функционировать при выключении основного источника питания (сеть 220 В). В этом случае устройство должно продолжать работать от аккумуляторов, поддерживая непрерывность передачи данных.

Выводы по разделу

В 1 разделе было сформирована цель работы, проанализированы аналоги и сравнены минусы и плюсы с разрабатываемой системой. Определены ключевые ключевые критерии для разработки устройства.

2 Конструкторское проектирование устройства

Конструкторское проектирование — это способ создания технического объекта, который заключается в определении технических требований к изделию и составлении рабочих чертежей. Конструкторское проектирование основывается на применении научных знаний, инженерного творчества и современных технологий. Конструкторское проектирование может быть направлено на создание новых или улучшение существующих технических объектов, таких как машины, оборудование, приборы, здания. Конструкторское проектирование имеет большое значение для развития науки, техники и экономики, так как способствует повышению качества, надежности, эффективности и безопасности технических объектов.

2.1 Формирование структурной схемы устройства

Для конструирования устройство, необходимо в первую очередь сформировать структурную схему устройства для дальнейшего подбора элементов.

Структурная схема разрабатываемого нами устройства приведена на рисунке 2.1. Она включает модуль Arduino Nano, модуль SIM800L, стабилизатор с регулируемым напряжением на микросхеме LM317 в зарядном блоке, АКБ и BMS.

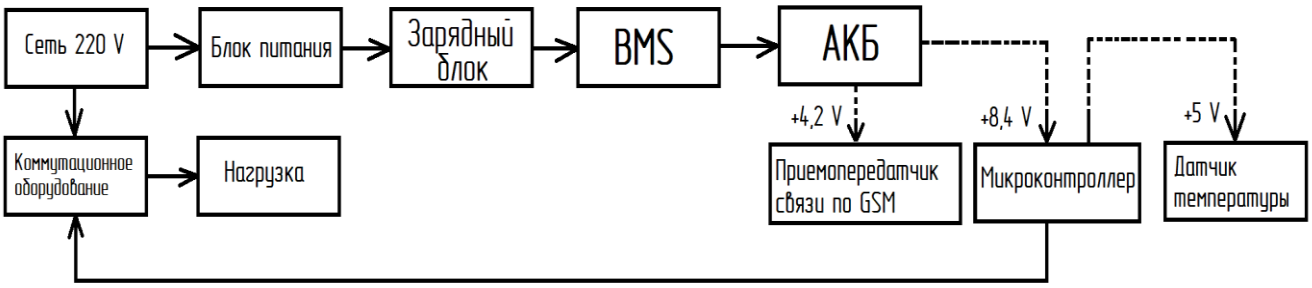

Рисунок 2.1 – Структурная схема устройства.

2.2 Анализ и выбор элементной базы для аппаратной части устройства

1) Выбор микропроцессорной части

Для реализации устройства мы выбрали платформу Arduino, которая является популярной и доступной для создания различных электронных проектов. Arduino Nano - это компактная и удобная плата, которая работает на микроконтроллере ATmega328. Устройство приведено на рисунке 2.2.

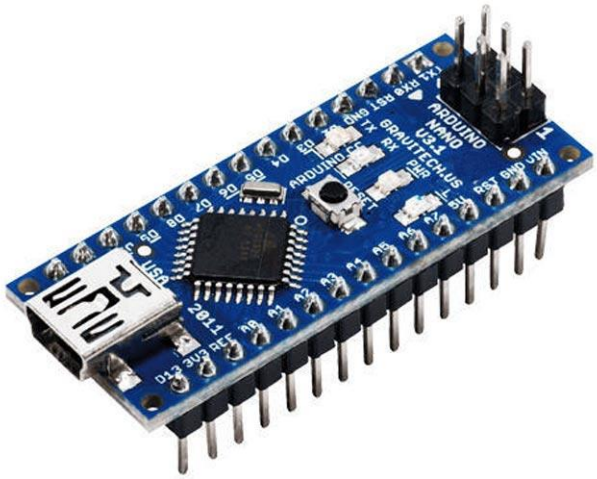

Рисунок 2.2 – Arduino Nano

Arduino Nano имеет 14 цифровых входов/выходов, из которых 6 могут использоваться как выходы ШИМ, и 8 аналоговых входов. Она также поддерживает интерфейсы UART, I2C и SPI для связи с другими устройствами [13]. Подробные характеристики приведены в таблице 2.

Таблица 2 – Характеристики микроконтроллера ATmega328

Параметр	Характеристика
Архитектура	AVR (Advanced Virtual RISC)
Тактовая частота	16 МГц
Цифровые входы/выходы	14 (включая 6 с возможностью работы в режиме PWM)

Продолжение таблицы 2

Оперативная память (RAM)	2 Кб
Флэш-память (программируемая память)	32 Кб (включая загрузчик)
EEPROM	1 Кб
Интерфейсы	USB (Micro-B), UART (Serial), I2C, SPI
Размеры	45 мм x 18 мм

Выбор Arduino Nano был обусловлен следующими причинами:

— Невысокая цена по сравнению с остальными Arduino платами, но не самое лучшее соотношение цена/возможности среди остальных Ардуино-совместимых плат на рынке;

— Широкий диапазон напряжений питания: стабильно работает от ~3 до 5 Вольт при питании “напрямую” (от 1.5 Вольт при понижении частоты процессора).

— Распаянный с завода стабилизатор напряжения для внешнего питания 7.. 15 Вольт

— Аппаратная поддержка самых популярных интерфейсов: UART, I2C, SPI

— Удобная в использовании, хорошо написанная официальная техническая документация для полноценной работы с МК и доступа ко всем его возможностям

- Быстрая компиляция и загрузка прошивки [14].

2) Выбор датчика температуры.

Далее нужно выбрать датчик для измерения температуры. Выбор пал на датчик DS18B20.

«DS18B20 – это полноценный цифровой термометр, способный измерять температуру в диапазоне от -55 °C до +125 °C с программируемой

точностью 9-12 бит. При изготовлении на производстве, каждому датчику присваивается свой уникальной 64-битный адрес, а обмен информацией с ведущим устройством (микроконтроллером или платой Arduino) осуществляется по шине 1-wire. Такой подход позволяет подключать к одной линии целую группу датчиков» [10]. Датчик представлен на рисунке 2.3.

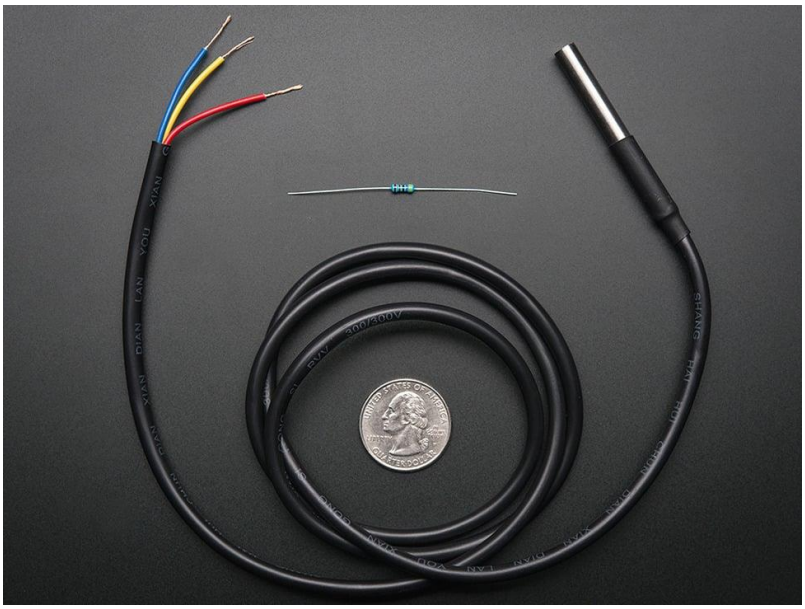

Рисунок 2.3 – Датчик температуры DS18B20

Он представляет собой небольшой и удобный в использовании датчик, который обеспечивает точные измерения температуры и имеет несколько преимуществ, важных для проекта домашней автоматизации:

— Цифровой интерфейс 1-Wire: Датчик использует одножильную шину данных для связи с микроконтроллером, что делает его удобным в использовании и экономит количество проводов для подключения.

— Есть возможность программирования диапазона тревожного сигнала.

— Простота подключения: Благодаря цифровому интерфейсу и простому протоколу обмена данными, DS18B20 легко интегрируется с микроконтроллерами, такими как Arduino, и обеспечивает быструю настройку и начало работы [10].

3) Выбор нагрузки.

В качестве нагрузки мы используем электрический водонагреватель. При выборе нагревателя учитывались следующие факторы:

— Мощность. От мощности нагревателя зависит его способность обогревать определённый объем воды. В среднем для обогрева 100 литров воды хватит 1,5 кВт.

— Цена. Нагреватель должен иметь хорошее соотношение цены и качества.

По данным критериям был найден электрический водонагреватель «EDISSON ER 100 V ЭдЭ001798» с мощностью 1,5 кВт [4]. Его вид приведен на рисунке 2.4.

Рисунок 2.4 – Электрический водонагреватель «EDISSON ER 100 V ЭдЭ001798».

4) Выбор силового коммутационного оборудования.

Для выбора реле для выбранного нагревателя, необходимо рассчитать ток, который должен коммутировать реле. Воспользуемся формулой 1.

$$I = \frac{P}{U}, \quad (1)$$

где P – мощность нагрузки, U – напряжение, в данном случае напряжение сети 220 В.

Рассчитаем ток, при котором будет совершена коммутация выбранного электрического водонагревателя с мощностью 1,5 кВт (формула 2).

$$I = \frac{P}{U} = \frac{1500}{220} = 6{,}8 \text{ А} \quad (2)$$

Получим, что реле должно выдерживать ток 6,8 А. Было решено, для безопасности, взять реле на 10 А.

Для управления нагревателем в данной работе будет использоваться реле SRD-05VDC-SL-C на 10 А. Оно представляет собой электромеханическое устройство, используемое для управления высоковольтными и высокотоковыми устройствами через микроконтроллеры или другие устройства. Реле приведено на рисунке 2.5.

Рисунок 2.5 – Реле SRD-05VDC-SL-C.

Вот основные особенности этого реле:

— Номинальное рабочее напряжение 5 В, эксплуатационный ток 71,4 мА, предельный коммутируемый ток 10 А, напряжение постоянного тока 30 В, переменного 250 В.

— Короткое время срабатывания: включение 10 мс, отключение 5 мс.

— Компактный размер и защитная оболочка, выдерживающая сильный нагрев, что предотвращает поломку содержимого [20].

4) Выбор приемопередатчика для работы устройства по GSM каналу

Для отправки СМС, звонков и передачи данных на удаленный сервер через технологию GPRS будем использовать модуль SIM800L.

SIM800L - это модуль GSM от Simcom, который предоставляет любому микроконтроллеру функциональность GSM/GPRS, он приведен на рисунке 2.6.

Рисунок 2.6 – GSM модуль SIM800L

Он может подключаться к мобильной сети для приема звонков и отправки и получения текстовых сообщений, а также подключаться к интернету с использованием GPRS, TCP или IP.

Характеристики SIM800L:

—Частотный диапазон: 850/900/1800/1900 МГц

— GPRS класс 12: Макс. 85.6 кбит/с

— Рабочее напряжение: 4.6-5.2V

— Уровень TTL: 2.9-5V

— Рабочая температура: -40 ~ 85°C

— Размеры: 40x28 мм [21].

SIM800L выбран по следущим причинам:

— Глобальная поддержка: SIM800L поддерживает четырехдиапазонные сети GSM/GPRS, что означает, что он будет работать почти в любой точке мира.

— Маленький размер и низкое энергопотребление.

Далее, необходимо выбрать тариф, так как сим-карта играет важную роль в системе. Рассмотрим различные варианты тарифных планов, а также проанализируем их преимущества и недостатки в таблице 3 [3],[6],[7],[9].

Таблица 3 – Сравнение тарифов для системы

Оператор	Тариф	Интернет, Мб	SMS, шт.	Звонки, мин.	Цена в месяц, Р
Tele2	Интернет для вещей	300	100	100	100
Beeline	Телематика	100	100	50	90
Megafon	Умные вещи	2028	100	30	100
МТС	Для устройств	1024	100	100	150

Изучив таблицу, мы видим, что лучшим выбором является Tele2, потому что его единственный недостаток — наименьший объём интернет-трафика. Однако для нашей системы достаточно 100 мегабайт в месяц.

5) Зарядный блок

Для того, чтобы обеспечить элементы устройства питанием, рассмотрим спроектированную схему, приведенную на рисунке 2.7.

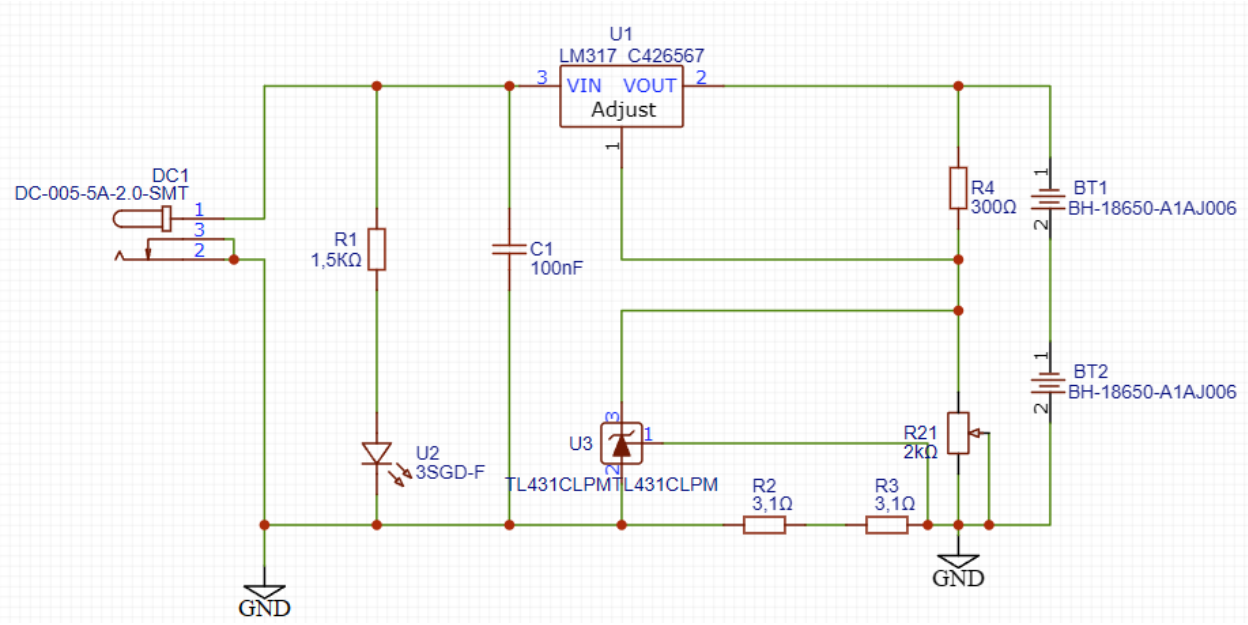

Рисунок 2.7 – Изображение схемы зарядного блока в программе EasyEda.

Состав схемы таков:

— LM317: это регулируемый стабилизатор напряжения с возможностью ограничения тока. Он широко используется в схемах зарядных устройств для аккумуляторов.

Микросхема работает так. Когда аккумулятор садится, она становится слабее по напряжению, чем источник питания. Если мы просто подключим источник питания к аккумулятору, то он может заряжаться слишком быстро и слишком сильно из-за большой разницы в силе. Из-за этого может случиться даже воспламенение. LM317, включенный по схеме ограничения тока с помощью обратной токовой связи, собранной на резисторах R19 и R20 контролирует ток зарядки, предотвращая перегрузку аккумулятора. Благодаря этому процесс зарядки становится безопасным и эффективным.

Характеристики LM317 следующие:

— Выходное напряжение: 1,2…37 В;

— Максимальный ток нагрузки : 1,5 А;

— Максимальное входное напряжение: 40 В [19].

— TL431: Это программируемый стабилизатор напряжения. Он работает как “управляемый стабилитрон”, сравнивая входное напряжение с опорным и открывая или закрывая выходной транзистор в зависимости от этого.

Характеристики TL431 следующие:

— Напряжение на выходе 2,5-36 В

— Опорное напряжение 2440-2550 мВ

— Отклонение опорного напряжения по полному температурному диапазону 4-25 мВ

— Прямой ток 1-100 мА

— Опорный ток 2-3 мкА

— Минимальный регулируемый ток катода 0,4-1 мкА Ток катода в выключенном состоянии 0,1-1 мкА [2].

Схема микросхемы приведена на рисунке 2.8

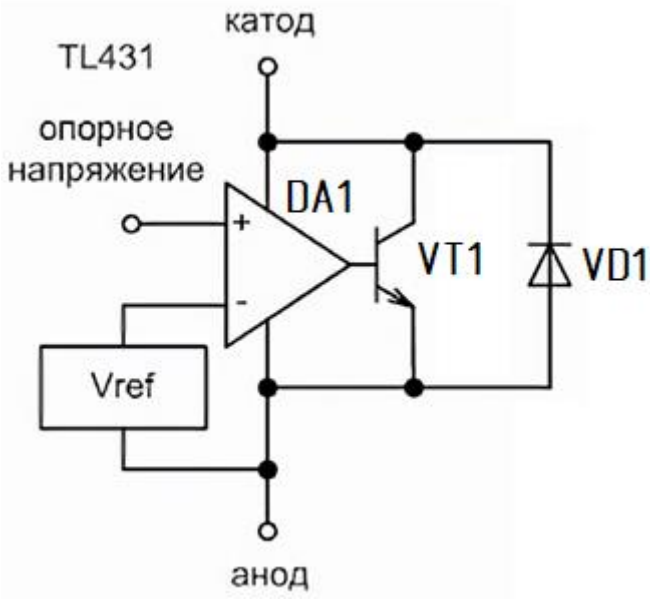

Рисунок 2.8 –Схема внутреннего устройства микросхемы TL431.

Алгоритм работы TL431 следующий. Операционному усилителю на вход стоит источник опорного напряжения на 2.5В, который подсоединен ко входу. Пин REF и коллектор и эмиттер транзистора связаны с контактами

питания усилителя. А безопасность обеспечивает защитный диод VD1, изображенный на рисунке 2.8. Он сохранит и убережет микросхему от переполюсовки.

Чтобы открылся выходной транзистор, нужно на вход REF подать сигнал, напряжение которого будет чуть больше, чем опорное. Так как достаточно превышения в пару милливольт, то смело можем считать, что подаем напряжение, которое равно опорному. В таком случае, на выходе с ОУ идет напряжение на базу транзистора, и он открывается. Получается, что эта микросхема работает как полевой транзистор. Она безостановочно сравнивает входное напряжение с опорным, и, когда напряжение на входе больше, она открывается, что позволяет АКБ заряжаться [22].

Рассчитаем мощность тепловыделения на LM317 (формула 3).

$$P = I_{\text{зар}} * (E_{\text{вх}} - 2 * U\text{акб}_{\min}) = 0{,}4 * (12 - 2 * 3{,}7) = 1{,}84 \text{ Вт}, \quad (3)$$

Где $U_{\text{акбmin}}$ – минимальное рабочее напряжение аккумуляторов, В;

$I_{\text{зар}}$ – ток зарядки, выходящий из BMS, мА;

$E_{\text{вх}}$ – входное напряжение схемы, В.

LM317 имеет корпус TO220. У этого корпуса максимальная допустимая мощность рассеивания 1 Вт [5]. Принято решение добавить к LM317 радиатора.

Рассчитаем сопротивление радиатора (формула 4).

$$R = \frac{\Delta T_{max}}{P} = \frac{80-40}{1{,}84} = 22 \ \frac{℃}{\text{Вт}}, \quad (4)$$

Где ΔT_{max} – разница между максимальной рабочей температуры и нормальной температуры, ℃;

P – мощность радиатора, Вт.

Получив значение сопротивления радиатора в 22 ℃/Вт, найдем нужный вариант. По необходимым техническим характеристикам, таким как сопротивление радиатора и малый размер, был выбран радиатор HS 201-30 с коэффициентом теплового сопротивления 19 ℃/Вт и длиной 30 мм [18]. Схематичный вид приведен на рисунке 2.9.

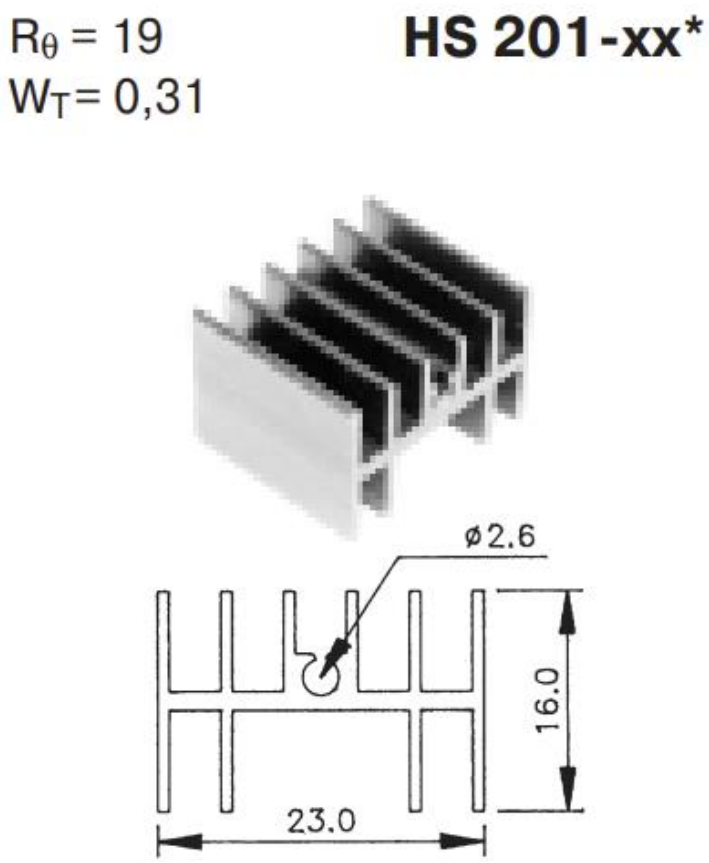

Рисунок 2.9 – Радиатор HS 201-30 для LM317.

В работе радиатор приведен в разделе о печатной плате, нарисованный схематично у LM317.

— Резисторы (R1, R2, R3, R4, R5, R6, R7, R8): Они использованы для установки опорного напряжения для TL431 и установки тока зарядки для LM317.

— Транзистор VT1 – в этой схеме он является ключом для переключения режимов работ зарядного блока.

— Конденсатор C1- используется для сглаживания выходного напряжения LM317.

— Диод HL1: HL1 может быть использован как индикатор окончания зарядки.

— Аккумулятор BAT: Это аккумулятор, который заряжается с помощью этой схемы.

5) Аккумуляторные батареи

Для выбора аккумуляторных батарей, сравним основные типы и сведем данные в таблицу 4, анализируя их достоинства и недостатки [8].

Таблица 4 – Сравнение типов аккумуляторов

Тип аккумулятора	Преимущества	Недостатки
Никель-кадмиевые	Низкая стоимость, Работа в широком диапазоне температур.	Высокая степень саморазряда, Большие размеры, малый срок службы, эффект «памяти»
Никель-металл-гидридные	Высокая энергетическая плотность.	Высокий саморазряд
Щелочные	Работоспособность на сильном морозе, долгий срок службы, устойчивость к избыточному заряду	Низкое напряжение, необходимость, эффект «памяти», требуют особого ухода
Литий-ионные	Высокая ёмкость, низкий саморазряд, долгий срок службы.	Относительная дороговизна, ухудшение работы при низких температурах.

Исходя из сравнения, литий-ионные аккумуляторы представляют собой самый оптимальный выбор для питания нашего устройства, так как они обладают высокой энергетической плотностью, низким саморазрядом, термической стабильностью, долгим сроком службы и безопасностью. Эти характеристики делают их идеальным выбором для устройств, которые требуют длительного времени работы и стабильной работы в обычных, не очень низких температурах.

В данной работе решено было выбрать литий-ионные аккумуляторы типоразмера 18650 с примерной ёмкостью 3500 мА*ч. Пример таких аккумуляторов приведен на рисунке 2.10.

Рисунок 2.10 – Литий – ионные аккумуляторы 18650.

6) BMS.

Система управления батареей (BMS) играет важную роль в проектах, связанных с аккумуляторами, особенно с литий-ионными. BMS предоставляет защиту от перезарядки и глубокого разряда: аккумуляторы могут быть повреждены, если они полностью разряжаются или перезаряжаются [12]. BMS контролирует процесс зарядки и разрядки, чтобы предотвратить эти ситуации. По этой причине введение BMS в проект обязательно.

Схема BMS, которая будет использоваться в работе, приведена на рисунке 2.11.

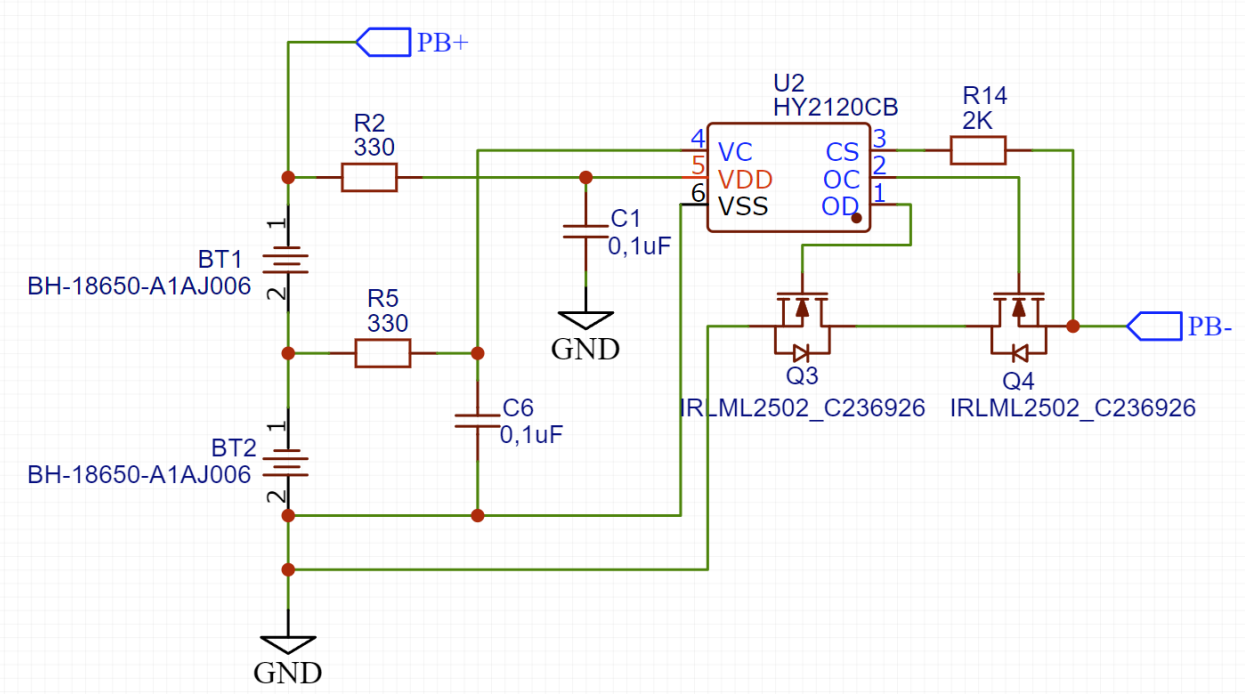

Рисунок 2.11 – Изображение схемы BMS в программе EasyEda.

Эта схема предназначена для управления двумя Li-ion аккумуляторами, соединенными последовательно. Вот как работают основные компоненты:

—Микросхема HY2120 – основной компонент схемы, он управляет зарядкой и разрядкой аккумуляторов и обеспечивает защиту от перезарядки и переразрядки.

— Транзисторы Q3 и Q4 – Используются как ключи для управления зарядкой и разрядкой аккумуляторов. Они подключают и отключают нагрузку и зарядку от аккумуляторов.

— Резисторы R2 и R5: Эти резисторы фильтра помех, который измеряет напряжения на АКБ.

— Конденсаторы C1 и C2: Эти конденсаторы защищают от всплесков напряжения.

— Резистор R14: Этот резистор разрешает работу микросхемы.

2.3 Анализ и выбор сервиса для мониторига температуры через Интернет

Сервисы для мониторинга температуры через Интернет - это специализированные платформы, которые позволяют подключать и управлять IoT-устройствами, такими как датчики, реле, нагреватели и другие. С их помощью можно получать данные о температуре в различных местах и условиях, а также регулировать параметры и функции системы отопления или кондиционирования воздуха. Сервисы для мониторинга температуры через Интернет могут быть полезны как для домашнего использования, так и для промышленных и коммерческих целей.

Однако, не все сервисы одинаковы. Они могут отличаться по функционалу, стоимости, удобству, надежности и безопасности. Поэтому, перед тем, как выбрать подходящий сервис для своего проекта, необходимо сравнить их между собой и оценить их преимущества и недостатки. Рассмотрим несколько популярных сервисов для мониторинга температуры через Интернет, таких как Blynk, Cayenne, ThingSpeak и Adafruit IO, и сравним их и сведем информацию в таблицу 5.

Таблица 5 – Сравнение IoT сервисов

Сервис	Гибкость настройки	Поддержка оборудования	Интерфейс	Готовые библиотеки	Дашборды, графики и виджеты	Цена
Blynk	Высокая, но завязанная на внутр. библиотеках	Любое оборудование	Приложение и веб интерфейс	Есть	Да	Бесплатно для личного использования
Cayenne	Низкая	Малая часть оборудования	Веб интерфейс	Нет	Да	Бесплатно для личного использования

Продолжение таблицы 5

ThingSpeak	Высокая	Популярное оборудование	Приложение и веб интерфейс	Есть	Да	Бесплатно для небольших проектов, платно для больших
Adafruit IO	Низкая	Arduino, ESP8266, Raspberry Pi	Приложение и веб интерфейс	Нет	Да	Бесплатно для небольших проектов, платно для больших проектов

Исходя из предоставленной информации, ThingSpeak выглядит как лучший выбор из-за своей гибкости, удобства использования и разнообразия функций.

2.4 Разработка электронной схемы

Для разработки электронной схемы выбран онлайн редактор принципиальных схем EasyEDA.

EasyEDA — кросс-платформенная веб-ориентированная среда автоматизации проектирования электроники включающая в себя редактор принципиальных схем, редактор топологии печатных плат, SPICE-симулятор, облачное хранилище данных, систему управления проектами, а также средства заказа изготовления печатных плат [15]. Этот сервис был выбран из-за простоты как моделирования схемы, так и разводки печатной платы.

Общая схема разделена на 2 функциональных каскада:

— каскад питания с BMS

— каскад логического взаимодействия

Электрическая принципиальная схема питания с BMS приведена на рисунке 2.12.

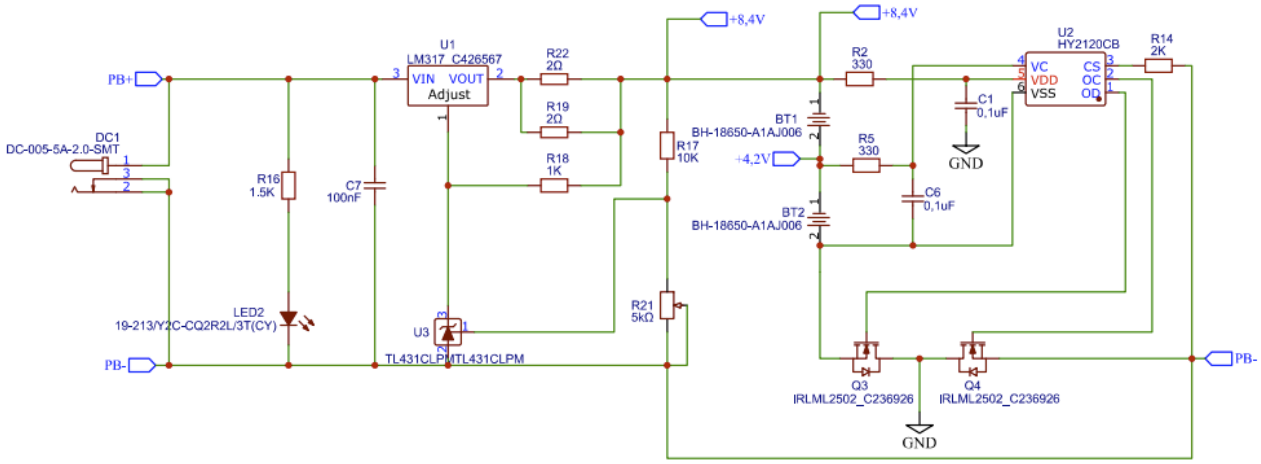

Рисунок 2.12 – Изображение схемы зарядного блока и BMS в программе EasyEda.

Для взаимодействия микроконтроллера с другими элементами создан каскад логического взаимодействия, включающий в себя микроконтроллер подключенными к нему цепью питания, реле, GSM модуль и датчик температуры. Этот каскад приведен на рисунке 2.13.

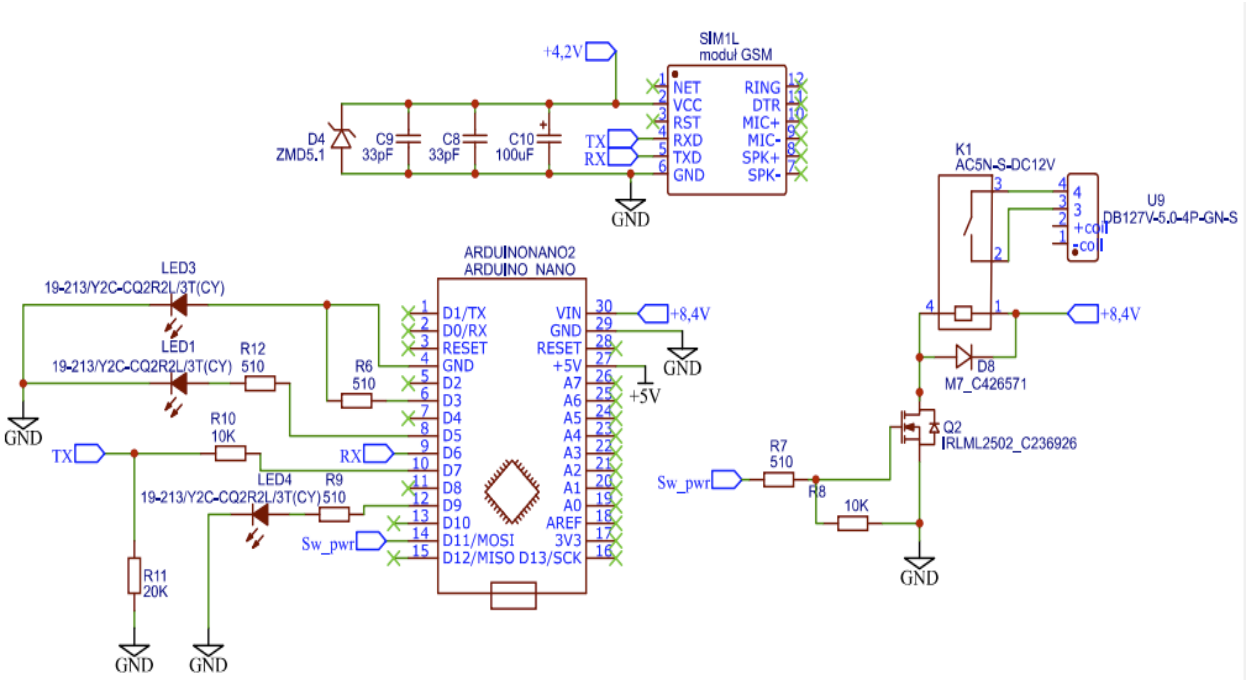

Рисунок 2.13 – Изображение схемы каскада логического взаимодействия в программе EasyEda.

2.5 Разработка печатной платы

После создания принципиальной схемы устройства, следующим шагом надо создать печатную плату. В EasyEDA предоставляет функционал преобразования принципиальной схемы в печатную плату, воспользуемся этой возможностью. Штатной функцией можно преобразовать схему в PCB формат. Это формат, содержаний информацию и расположение дорожек, электронных компонентов и технических отверстий. Результат разводки платы приведен на рисунке 2.14.

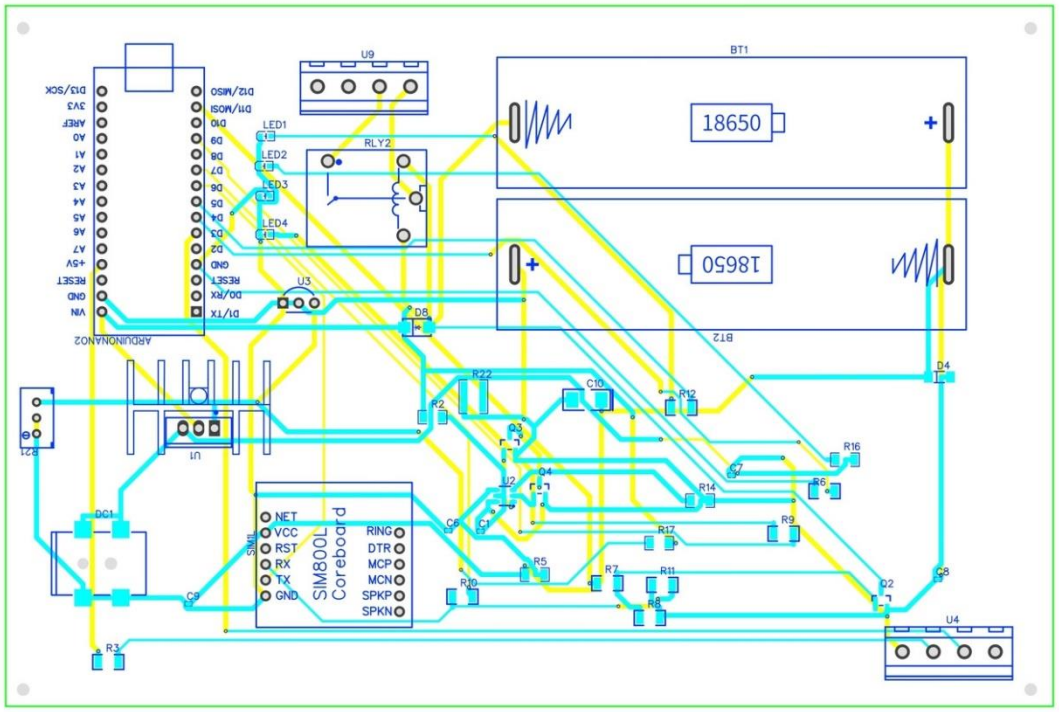

Рисунок 2.14 – Печатная плата устройства.

Выводы по разделу

Во 2 разделе была сформирована структурная схема. По ней были подобраны все элементы. Кроме того, был подобран веб-сервис для мониторинга температуры через Интернет, а так же разработана электронная схема и печатная плата устройства.

3 Программирование устройства

Программирование устройства на базе Arduino - это многоэтапный процесс, который начинается с тщательной подготовки и заканчивается тестированием и документированием. Программирование включает в себя несколько шагов:

1) Конфигурация параметров: Необходимо настроить параметры для подключения к сети GSM, включая APN, которые обеспечат доступ к мобильной сети. Также следует правильно настроить пины на плате Arduino для подключения датчика температуры и GSM модуля, чтобы обеспечить их корректное взаимодействие.

2) Инициализация компонентов: После настройки параметров следует инициализировать датчик температуры, реле и GSM модуль. Это включает в себя установку соединения с сетью и проверку работоспособности модуля, чтобы убедиться, что он готов к отправке и приему данных.

3) Программирование логики работы: На этом этапе пишется основная логика работы устройства. Это включает функцию для считывания температуры с датчика и отправки этих данных на сервер ThingSpeak, а также функции для отправки SMS и совершения звонков при определенных условиях, например, когда температура опускается ниже заданного порога. Кроме того, нужно прописать логику для включения нагревателя через реле при определенном температурном пороге.

4) Тестирование и отладка: После программирования следует загрузить код на Arduino и провести тестирование всех компонентов системы. Если в процессе тестирования обнаруживаются ошибки или непредвиденное поведение системы, необходимо провести отладку кода.

3.1 Аппаратная часть

Аппаратная часть системы представляет собой плату Arduino Nano, используемую для отладки. Плата Arduino является инновационной и доступной средой для проектирования и кодирования электронных механизмов. Она базируется на C++ и включает в себя комплекс аппаратных средств и программного обеспечения, что делает её идеальной для разработки функциональных приложений. Благодаря упрощённому языку программирования, подходящему для начинающих, и возможности простого соединения с различными электронными компонентами через встроенные интерфейсы и порты, плата Arduino обеспечивает гибкость и удобство в использовании.

Кроме того, плата Arduino Nano обладает компактными размерами, что делает её идеальным выбором для проектов с ограниченным пространством. Её многофункциональность и модульность позволяют легко интегрировать её в разнообразные проекты, от простых устройств до сложных автоматизированных систем. Arduino Nano становится мостом между традиционной электроникой и цифровым миром, открывая новые горизонты для творчества и инноваций.

3.1.1 Описание функций работы аппаратной части

Аппаратная часть системы отвечает следущим требованиям, предъявляемые к функционалу:

1) Установление соединения с серверной частью сервиса ThingSpeak, реализуемое посредством технологии GSM.

GSM имеет ряд преимуществ по сравнению с Wi-Fi и 3G/4G:

Покрытие: GSM обеспечивает более широкое покрытие по сравнению с Wi-Fi, что делает его более подходящим для IoT-проектов, которые требуют передачи данных на большие расстояния.

Надежность: GSM является более надежной технологией по сравнению с Wi-Fi, особенно в условиях, когда Wi-Fi-сигнал может быть подвержен помехам.

Энергоэффективность: Устройства GSM обычно потребляют меньше энергии по сравнению с устройствами Wi-Fi и 3G/4G, что делает их более подходящими для IoT-устройств, работающих от батареи.

Стоимость: Во многих случаях стоимость использования GSM может быть ниже, чем стоимость использования 3G/4G, особенно для передачи небольших объемов данных.

2) Мониторинг температуры в помещении

3) Оповещение абонента о критическом понижении температуры посредством СМС сообщения

4) Оповещение абонента о критическом понижении температуры посредством звонка на номер телефона

5) Переключение реле, которое включает аварийный нагреватель при понижении температуры до критической отметки

3.1.2 Описание программы и алгоритма аппаратной части

Для того чтобы начать писать программу, необходимо из технических требований, предъявляемых в тексте, создать графическое представление алгоритма работы будущей программы. Необходимо создать блок-схему алгоритма, она представлена на рисунке 3.1.

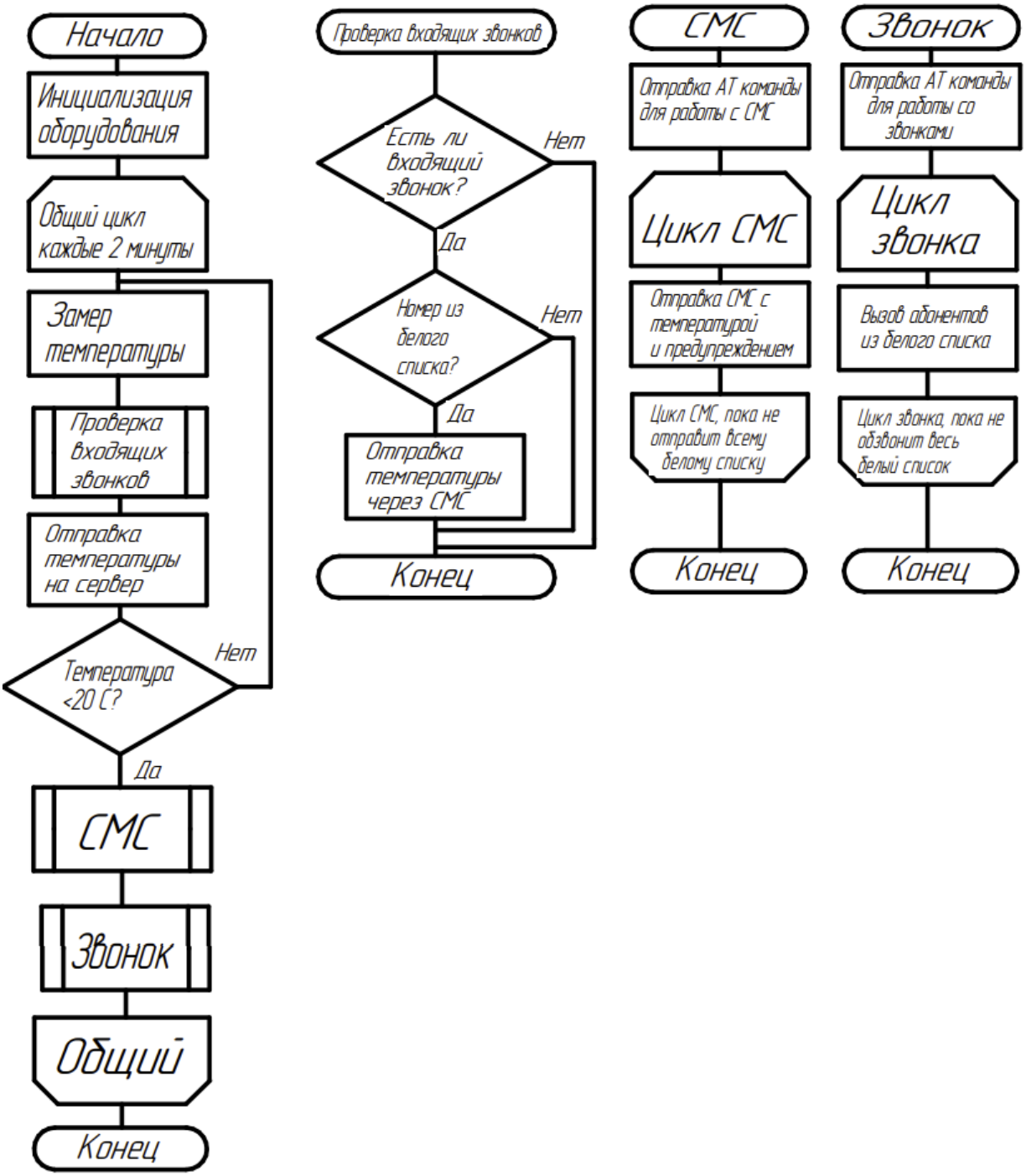

Рисунок 3.1 – Блок схема алгоритма.

Исходя из функционала, описанного ранее и блок-схемы алгоритма, разработана программа для соответствия всем требованиям. Для написания программы для Arduino мы использовали язык Arduino C, который основан на C/C++. Код делится на три части: подготовка, настройка (setup) и цикл (loop). В каждой части используются разные библиотеки, константы,

переменные, функции и команды для работы с датчиком температуры и GSM модулем. Полный код с комментариями приведен в приложении А.

В части подготовки программного кода к работе необходим ряд настроек:

1) Подключение библиотек. Здесь мы подключаем все необходимые библиотеки. OneWire и DallasTemperature - для работы с датчиками температуры Dallas, а SoftwareSerial - для создания дополнительного серийного порта.

2) Определение пинов оборудования. Тут прописываем пины, которые будут использоваться для подключения датчика температуры (ONE_WIRE_BUS) и реле (RELAY_PIN).

3) Создание обьектов. В этом блоке мы создаем объект oneWire класса OneWire, который используется для взаимодействия с датчиками, подключенными по шине OneWire. В качестве параметра мы передаете пин, к которому подключен датчик (ONE_WIRE_BUS). Далее создаем объект sensor класса DallasTemperature, который используется для взаимодействия с датчиками температуры Dallas. В качестве параметра передаем ссылку на объект oneWire, который будет использоваться для общения с датчиками.

4) Создаем объект gprsSerial. это часть класса SoftwareSerial. Это нужно для того, чтобы сделать второй серийный порт. 6 и 7 - это номера пинов RX и TX

5) Создание переменной tempC типа float. Мы будем использовать её, чтобы хранить последнее измеренное значение

7) Добавление белого списка контактов а также получение информации о размере списка.

Все выше сказанное приводится в листннге 1.

Листинг 1

```
#include <SoftwareSerial.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 3
#define RELAY_PIN 4
```

```
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
SoftwareSerial gprsSerial(6,7);
float tempC;
String whiteList[] = {"+79060000000"};
int numNumbers = sizeof(whiteList) / sizeof(whiteList[0]);
```

Теперь разберем функции, которые приведены в блоке подготовки.

1) void sendSMS(String number, String message)

Функция sendSMS(String number, String message) - это способ отправки SMS-сообщений. Она берет два вещи: номер телефона и текст сообщения. Затем она отдает набор команд модулю SIM800. Эти команды делают следующее: устанавливают формат сообщения как текстовый (AT+CMGF=1\r), указывают номер телефона, на который нужно отправить сообщение (AT+CMGS="" + number + ""\r), вводят текст сообщения и отправляют сообщение с помощью символа подтверждения ASCII (26).

2) callNumber(String number): Эта функция используется для совершения звонка. Она принимает один параметр: номер телефона. Функция отправляет команду модулю SIM800 для набора номера (ATD + number + ;), а затем через 30 секунд завершает вызов (ATH).

3) checkIncomingCall ((float tempC) проверяет входящие звонки. Она начинается с цикла while, который выполняется, пока есть доступные данные для чтения из последовательного порта GPRS. Эти данные считываются и сохраняются в переменной incomingData.

Затем функция проверяет, содержит ли incomingData строку “+CLIP:”, которая является уведомлением о входящем звонке от GSM-модуля. Если “+CLIP:” найден, функция начинает просматривать белый список номеров.

Для каждого номера в белом списке функция проверяет, содержит ли incomingData этот номер. Если номер найден, функция выводит сообщение в монитор порта сообщение о том, что входящий звонок идет от номера из белого списка, а также отправляет на этот номер актуальную температуру.

Затем функция делает паузу в 3 секунды. Это сделано для предотвращения быстрой обработки нескольких входящих звонков подряд.

Все описанные функции в коде приведены в листинге 2.

Листинг 2

```
void sendSMS(String number, String message) {
gprsSerial.print("AT+CMGF=1\r");
delay(1000);
gprsSerial.print("AT+CMGS=\"" + number + "\"\r");
delay(1000);
gprsSerial.print(message);
delay(1000);
gprsSerial.print((char)26);
delay(1000);
gprsSerial.println();
Serial.println("Сообщение отправлено");
delay(1000);
}
void callNumber(String number) {
gprsSerial.println("ATD" + number + ";");
Serial.println("Аварийный звонок совершен");
delay(30000);
gprsSerial.println("ATH");
}

void checkIncomingCall(float tempC) {
while (gprsSerial.available()) {
String incomingData = gprsSerial.readString();
if (incomingData.indexOf("+CLIP:") != -1) {
for (int i = 0; i < numNumbers; i++) {
if (incomingData.indexOf(whiteList[i]) != -1) {
Serial.println("Звонок от номера из белого списка");
sendSMS(whiteList[i], "Actual temp" + String(tempC));
delay(3000);
}
}
}
}
}
```

Далее, разберем часть настройки (setup) кода.

1) setup(): Эта функция вызывается один раз при запуске программы. Она выполняет следующие действия:

- Инициализирует серийный порт со скоростью 9600 бод. Это необходимо для обмена данными между Arduino и SIM800.
- Запускает модем SIM800 со скоростью 9600 бод.

- инициализирует датчик температуры DallasTemperature, подключенный к Arduino.
- Отправляет команду «AT», которая проверяет общую работу модуля и выводит результат в монитор порта.
- отправляет команду “AT+CLIP=1” в GPRS модуль через последовательное соединение и выводит результат. Команда “AT+CLIP=1” включает функцию Caller Line Identification Presentation (CLIP) в модуле, что позволяет модулю узнавать номер входящего вызова.

Часть настройки (setup) кода приведена в листинге 3.

Листинг 3

```
void setup()
{
  gprsSerial.begin(9600);
  Serial.begin(9600);
  sensors.begin();
  delay(1000);

  gprsSerial.println("AT"); // Отправка команды AT
  delay(1000);
  if (gprsSerial.find("OK")) {
    Serial.println("Команда AT выполнена успешно");
  } else {
    Serial.println("Ошибка команды AT");
  }
delay(1000);
  gprsSerial.println("AT+CLIP=1"); // Включение CLIP
  delay(1000);
  if (gprsSerial.find("OK")) {
    Serial.println("Команда AT+CLIP=1 выполнена успешно");
  } else {
    Serial.println("Ошибка команды AT+CLIP=1");
  }
}
```

В заключение разбора кода, рассмотрим часть кода цикла (loop).

Этот код выполняется в бесконечном цикле, что означает, что он будет повторяться до тех пор, пока устройство не будет выключено. В каждом цикле код выполняет следующие действия:

1) Запрашивает температуру с датчика DS18B20 (sensors.requestTemperatures) и сохраняет температуру в переменной tempC (float tempC = sensors.getTempCByIndex(0)). Индекс «0» тут обозначает номер датчика, поскольку в программировании индексация начинается с нуля.

2) Вызывает функцию checkIncomingCall(), которая проверяет наличие новых входящих звонков

3) Отправляет различные AT – команды в GPRS модуль для настройки соединения.

4) Собирает HTTP GET запрос для обновления температуры на сервере, где field1 – указание, в какое поле добавить новое значение.

5) gprsSerial.println(str) отправляет созданный GET-запрос в GPRS модуль для отправки на сервер ThingSpeak.

6) Отправляет символ Ctrl+Z (ASCII 26) в GPRS модуль, что сигнализирует о конце отправки данных.

7) Отправляет AT команду для закрытия GPRS соединения (gprsSerial.println("AT+CIPSHUT")).

8) Вызывает функцию checkIncomingCall() для проверки входящих звонков после отправки данных на сервер.

9) Делает задержку в 60 000 миллисекунд, или одну минуту, перед следующим циклом.

10) Следующий блок кода с ветвлением, если была снята температура ниже 20 градусов, то система отправляет СМС оповещения всему белому списку, а позже звонит каждому номеру.

В листинге 4 приведена часть кода цикла (loop).

Листинг 4

```
void loop()
{

sensors.requestTemperatures();
```

```
float tempC = sensors.getTempCByIndex(0);
Serial.print("Текущая температура: ");
Serial.println(tempC);
checkIncomingCall(tempC);
gprsSerial.println("AT");
delay(1000);
gprsSerial.println("AT+CPIN?");
delay(1000);
gprsSerial.println("AT+CREG?");
delay(1000);
gprsSerial.println("AT+CGATT?");
delay(1000);
gprsSerial.println("AT+CIPSHUT");
delay(1000);
gprsSerial.println("AT+CIPSTATUS");
delay(2000);
gprsSerial.println("AT+CIPMUX=0");
delay(2000);
gprsSerial.println("AT+CSTT=\"airtelgprs.com\"");
delay(1000);
gprsSerial.println("AT+CIICR");//bring up wireless connection
delay(3000);
gprsSerial.println("AT+CIFSR");//get local IP adress
delay(2000);
gprsSerial.println("AT+CIPSPRT=0");
delay(3000);
gprsSerial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");
delay(6000);
gprsSerial.println("AT+CIPSEND");
delay(4000);

String str = "GET
https://api.thingspeak.com/update?api_key=VVS7LH6YT75J3YOO&field1=0" +
String(tempC);
Serial.println(str);
gprsSerial.println(str);
delay(4000);
gprsSerial.println((char)26);
delay(5000);
gprsSerial.println();
gprsSerial.println("AT+CIPSHUT");
delay(300);
checkIncomingCall(tempC);
delay(60000);

if (tempC < 20.0) {
for (int i = 0; i < numNumbers; i++) {
sendSMS(whiteList[i], "Warning: Temperature is below 20 degrees Celsius!");
delay(3000); // Ждем минуту
callNumber(whiteList[i]);
```

3.2 Серверная часть

Серверная часть проекта играет ключевую роль в обработке и управлении данными, полученными от аппаратной части. Она обеспечивает надежное и эффективное взаимодействие между пользовательским интерфейсом и аппаратной частью, обрабатывая данные от датчиков и отправляя команды обратно на устройство.

В данном проекте серверная часть будет реализована с использованием ThingSpeak. Это облачная платформа, предназначенная для разработки и управления проектами IoT. Она предоставляет простой и удобный интерфейс, который позволяет пользователям легко создавать приложения для управления и мониторинга различных IoT устройств.

В целом, серверная часть позволит нам максимально эффективно использовать возможности Arduino для мониторинга температуры, обеспечивая надежное и удобное взаимодействие между устройством и пользователем.

3.2.1 Описание функций работы серверной части

Серверная часть системы отвечает следующим требованиям, предъявляемые к функционалу:

1) Установление и поддержание стабильного соединения с аппаратной частью через сервис ThingSpeak. Серверная часть должна быть способна обрабатывать данные, полученные от аппаратной части.

2) Обработка и анализ данных о температуре, полученных от аппаратной части. Серверная часть должна быть способна обрабатывать некий объем данных в реальном времени.

3) Предоставление пользовательского интерфейса для мониторинга текущего состояния системы и управления ею. Серверная часть должна предоставлять простой и интуитивно понятный интерфейс, который позволит пользователю легко отслеживать состояние системы в реальном времени

3.2.2 Описание настройки серверной части

Для настройки серверной части нужно:

1) Создать аккаунт в ThingSpeak:

Сначала посетим официальный сайт ThingSpeak и пройдем классическую регистрацию.

2) Создать новый проект и создадим поля

Откроем главную страницу (щелкнув ПКМ на логотип) и нажмем на кнопку “ New Channel”. Дадим проекту название и описание.

Далее, назовем поля, с которыми будем работать. В нашем случае это «Temperature» (рисунок 3.2).

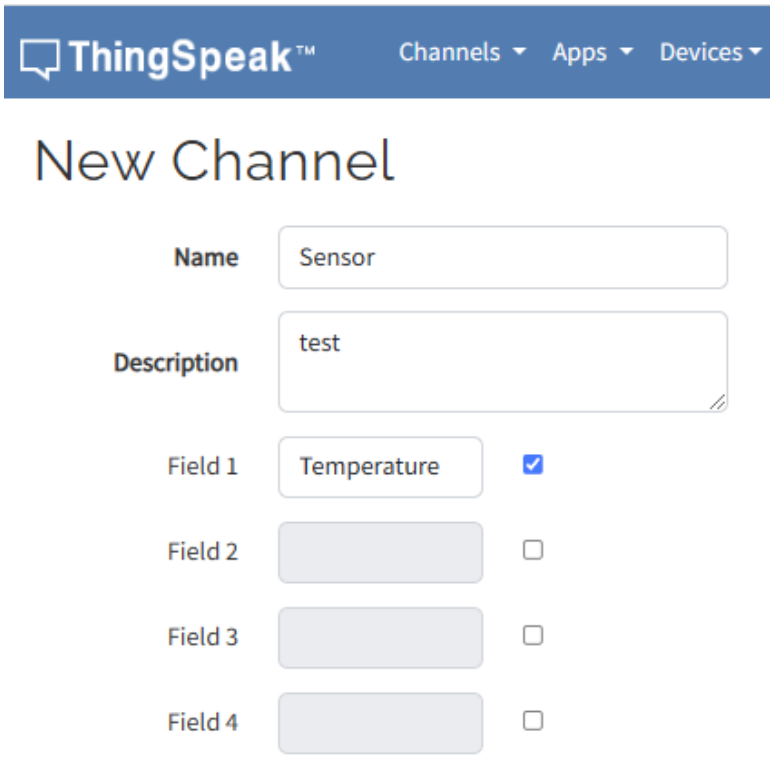

Рисунок 3.2 – Создание проекта и рабочего поля

3) Добавление необходимых виджетов.

Сервис предоставляет возможность выбрать множество виджетов для отображения информации, от самых простых, такие как обычный график на отображения одного параметра, до сложных, которые пользователь может

сам написать, используя весь функционал MATLAB. На рисунке 3.3 приведен пример кода на MATLAB, на рисунке 3.4 его реализация.

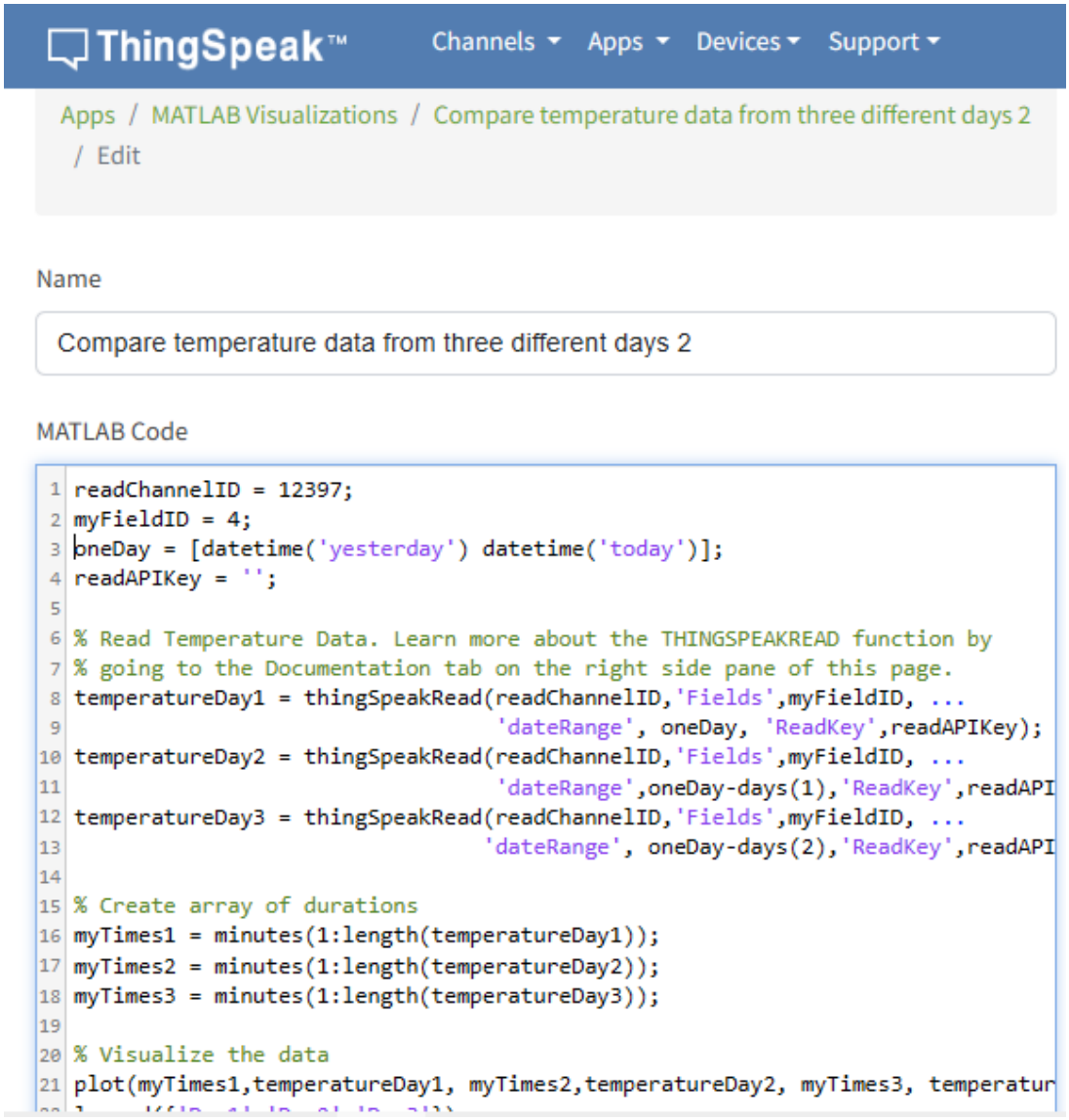

Рисунок 3.3 – Код примера отображения графиков в ThingSpeak.

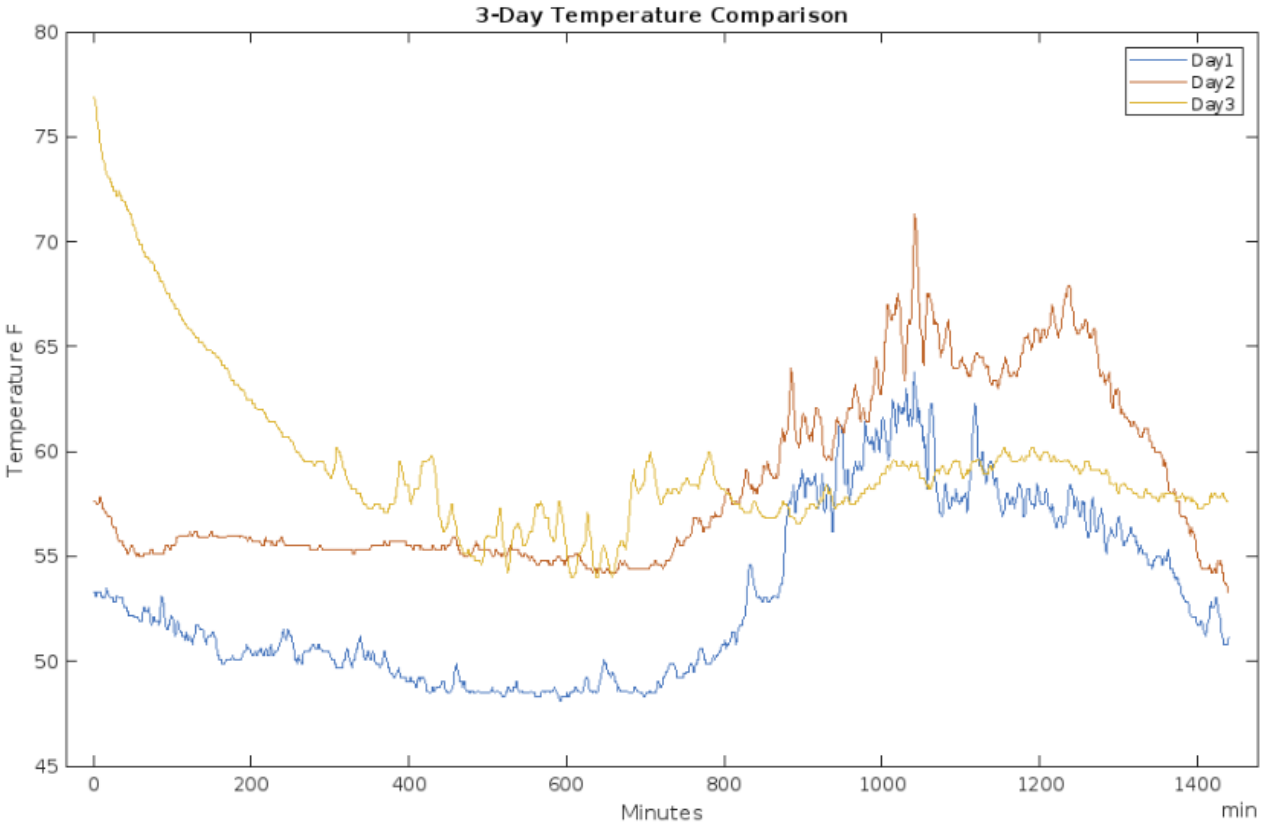

Рисунок 3.4 – Пример отображения графиков при помощи написания кода MATLAB

Поскольку нам необходимо мониторить только 1 параметр, было решено выбрать обычный график, а также виджет под названием «Gauge» (рисунок 3.5)

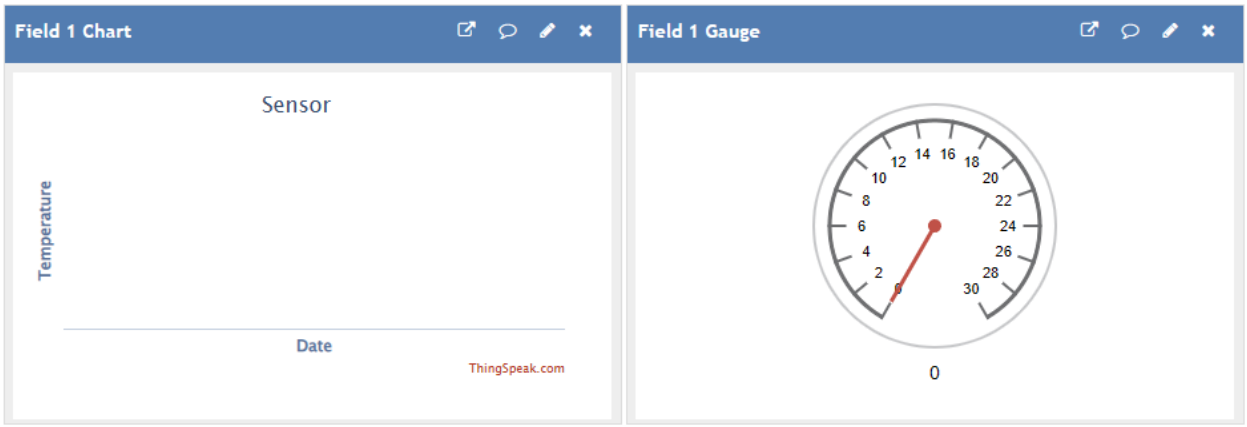

Рисунок 3.5 – График и виджет до настройки.

График настраивать нет необходимости, а вот виджет надо настроить, сделаем это.

На рисунке 3.6 приведено окно настройки «gauge». В нем разграничиваем табло на сектора разных цветов, а также меняем время обновления и шаг табло.

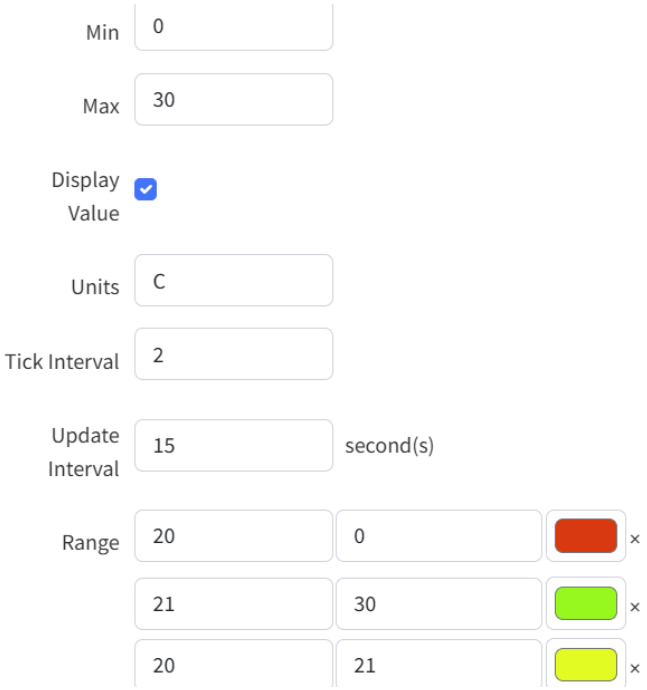

Рисунок 3.6 – Настройка виджета «Gauge».

На рисунке 3.7 можно увидеть рабочую область сервиса ThingSpeak после настройки, уже показана работа устройства.

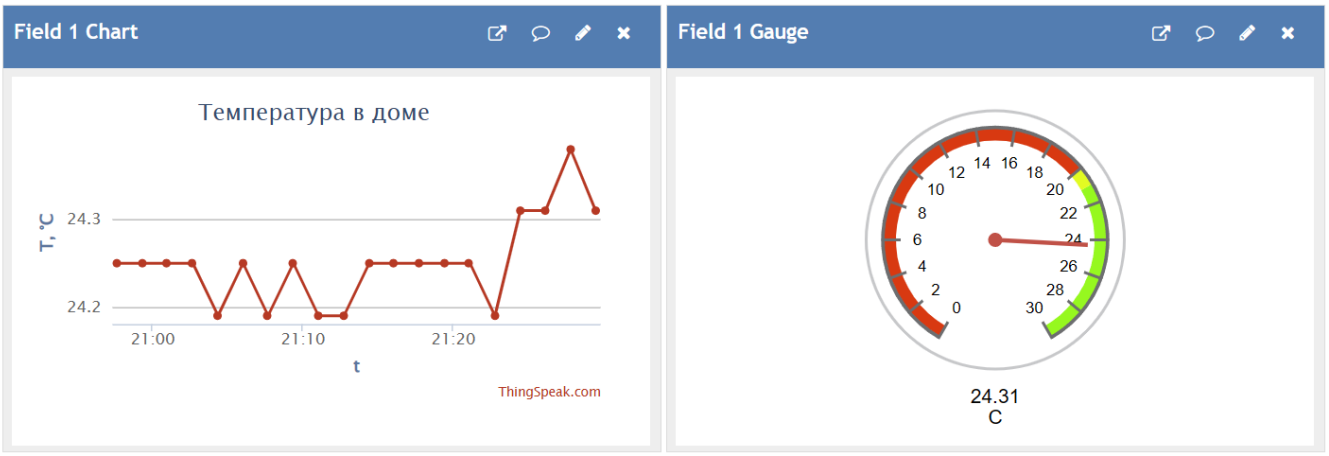

Рисунок 3.7 – График и виджет после настройки.

4) Получение необходимых данных для программирования Arduino.

После настройки и создания виджетов, приступаем к программированию Arduino. Для этого нужно получить токен, их можно найти в проекте во вкладке «API Keys», окно с токенами преведены на рисунке 3.8. Для передачи данных на сервер, нужен «Write API Key», его мы скопируем и вставим в код программы.

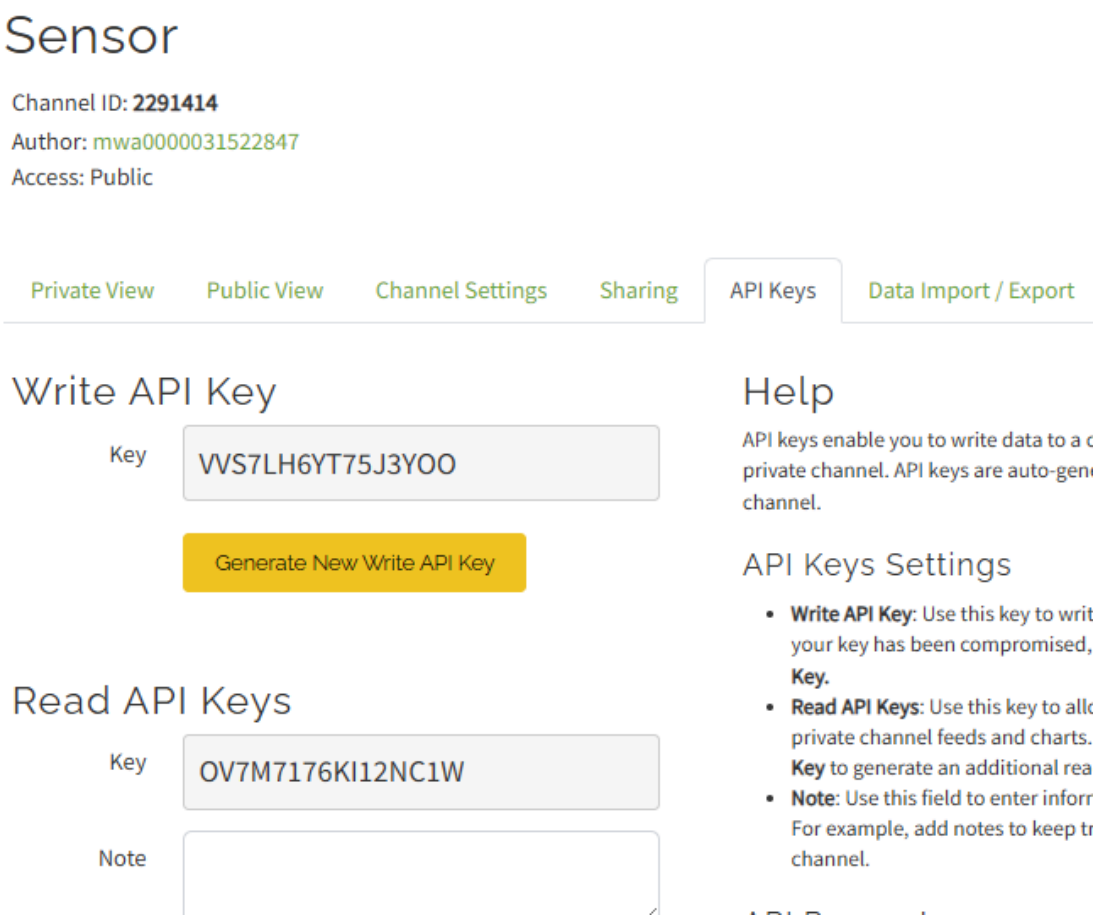

Рисунок 3.8 – Токены от сервиса ThingSpeak.

Выводы по разделу

В 3 разделе была сформирована блок-схема алгоритма и по ней написан код на Arduino. Также был настроен веб-сервис ThingSpeak для работы с разрабатываемом программно-аппаратным комплексом.

4 Экспериментальная часть проектируемой системы

С учётом всех предыдущих шагов, была собрана система (рисунок 4.1).

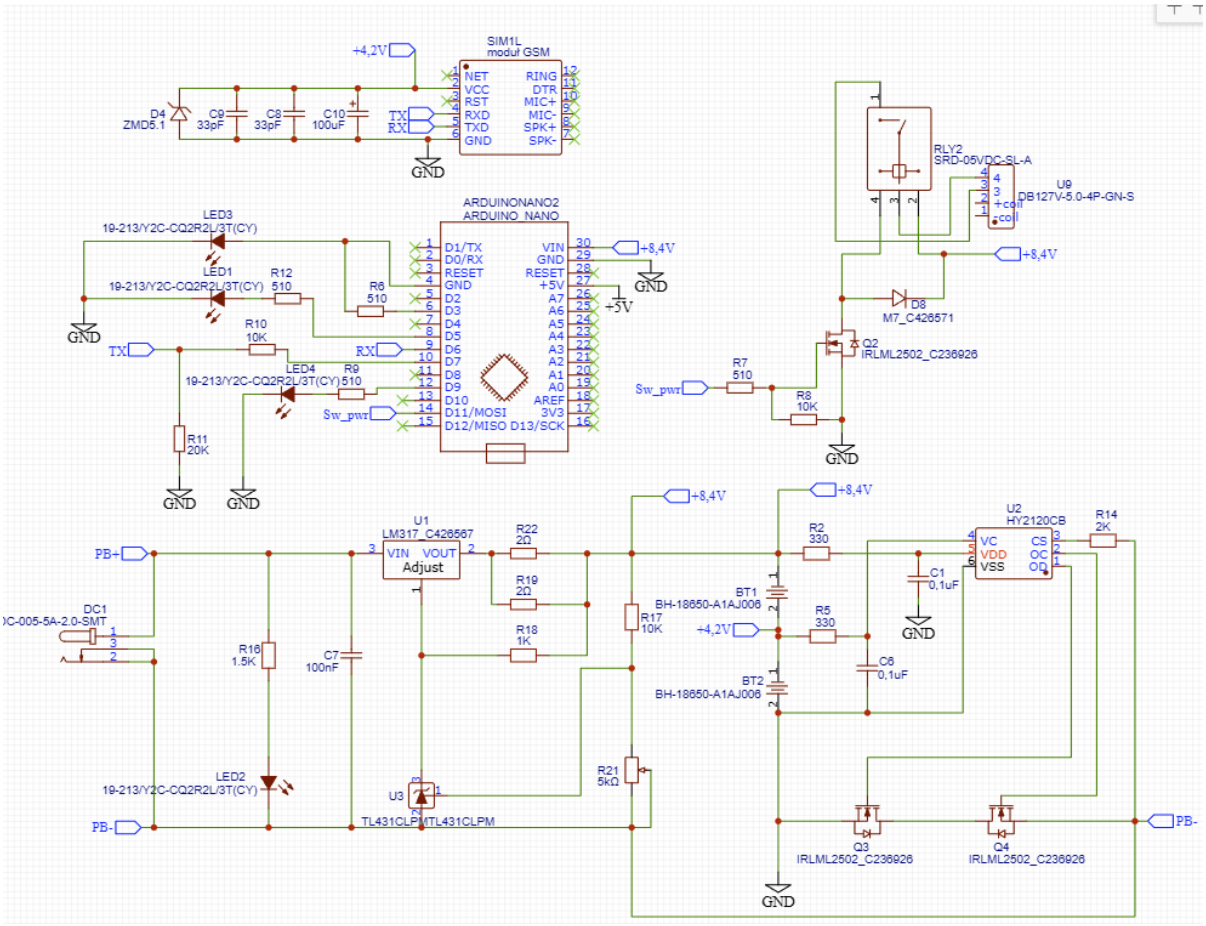

Рисунок 4.1 – Изображение схемы разработанного устройства в программе EasyEda.

Далее необходимо провести один из самых важных жизненных этапов любой разработки, а именно тестирования. Тестирование будет двухэтапное:

1) Тестирование аппаратного комплекса. В этом этапе будет проверка работы таких функций как переключение реле при понижении ниже критичной температуры, СМС и оповещние посредством звонка. Кроме того, будет проверена функция передачи актуальной температуры через СМС, если на систему позвонить.

2) Тестирование работы системы в сервисе ThingSpeak. В этом этапе будет проверка работы передачи актуальной температуры и построения графиков для наглядного отображения информации. Тестировать будем в

браузере на персональном компьютере, а также в приложении ThingShow на мобильном устройстве на базе Android.

4.1 Тестирование аппаратного комплекса

После сборки системы и написания программы под неё, было решено провести отладку системы в целом. Для этого будем использовать Serial порт как источник информации. Проверка работоспособности производилась поэтапно:

1) Произведено включение системы (рисунок 4.2).

```
20:27:31.738 -> Команда AT выполнена успешно
20:27:33.765 -> Команда AT+CLIP=1 выполнена успешно
20:27:34.292 -> Текущая температура: 25.19
20:28:03.579 -> GET https://api.thingspeak.com/update?api_key=VVS7LH6YT75J3YOO&field1=025.19
20:29:14.766 -> Текущая температура: 25.00
20:29:44.077 -> GET https://api.thingspeak.com/update?api_key=VVS7LH6YT75J3YOO&field1=025.00
20:30:55.286 -> Текущая температура: 24.87
20:31:24.578 -> GET https://api.thingspeak.com/update?api_key=VVS7LH6YT75J3YOO&field1=024.88
20:32:36.679 -> Текущая температура: 24.87
20:33:05.002 -> GET https://api.thingspeak.com/update?api_key=VVS7LH6YT75J3YOO&field1=024.88
```

Рисунок 4.2 – Ответ модуля на включение.

Как видно на рисунке 4.2, модуль посылает на запрос AT (проверка связи) положительный ответ, обозначающий, что связь установлена корректно. Кроме того, модулю посылается команда AT+CLIP=1, которая включает функцию определителя входящих номеров (АОН), на что модуль отвечает, что все прошло успешно.

2) Произведено понижение температуры в помещении (рисунок 4.3).

```
14:43:19.946 -> Текущая температура: 24.69
14:43:49.254 -> GET https://api.thingspeak.com/update?api_key=VVS7LH6YT75J3YOO&field1=024.69
14:45:00.469 -> Текущая температура: 24.75
14:45:29.770 -> GET https://api.thingspeak.com/update?api_key=VVS7LH6YT75J3YOO&field1=024.75
14:46:41.112 -> Текущая температура: 17.31
14:47:10.407 -> GET https://api.thingspeak.com/update?api_key=VVS7LH6YT75J3YOO&field1=017.31
14:48:25.191 -> Сообщение отправлено
14:48:29.239 -> Аварийный звонок произведен
```

Рисунок 4.3 – Ответ системы на понижение температуры ниже порога 20 градусов

Как видно из рисунка 4.3, при понижении температуры ниже 20 градусов в Serial порт выводится информационные сообщения «Сообщение отправлено» и «Аварийный звонок произведен». Это означает, что система уже отправила СМС на указанные в белом списке номера, а также позвонила на них. На рисунке 4.4 можно увидеть пример СМС.

Рисунок 4.4 – СМС, оповещающая о критической температуре.

3) Звонок на модуль системы SIM800L (рисунок 4.4).

```
Текущая температура: 22.00
RING

+CLIP: "+79061[redacted]",145,"",0,"",0
Text Sent.
ATH

OK

>
NO CARRIER
```

Рисунок 4.4 – Ответ системы на входящий звонок модулю

На рисунке 4.4 видим, что модуль замечает звонок (RING). Далее выводит сообщение +CLIP, где записан входящий номер. Этот номер сравнивается с

белым списком. Если номер находится в белом списке, модуль принимает звонок (ATA), и сбрасывает его (ATH). После этого, на этот входящий номер присылается текущая температура (рисунок 4.5).

Рисунок 4.5 – Принятая СМС, оповещающая о текущей температуре по запросу в виде входящего звонка.

4.2 Тестирование работы системы в сервисе ThingSpeak

После успешного тестирования основных функций системы, протестируем работоспособность функций, которые предоставляет сервис ThingSpeak. Для начала на ПК.

В личном кабинете проекта во вкладе «Dashboard» можно заменить график, построенный по значениям температуры, снятой с датчика температуры, представленой в виде соединеных между собой точек (рисунок 4.6).

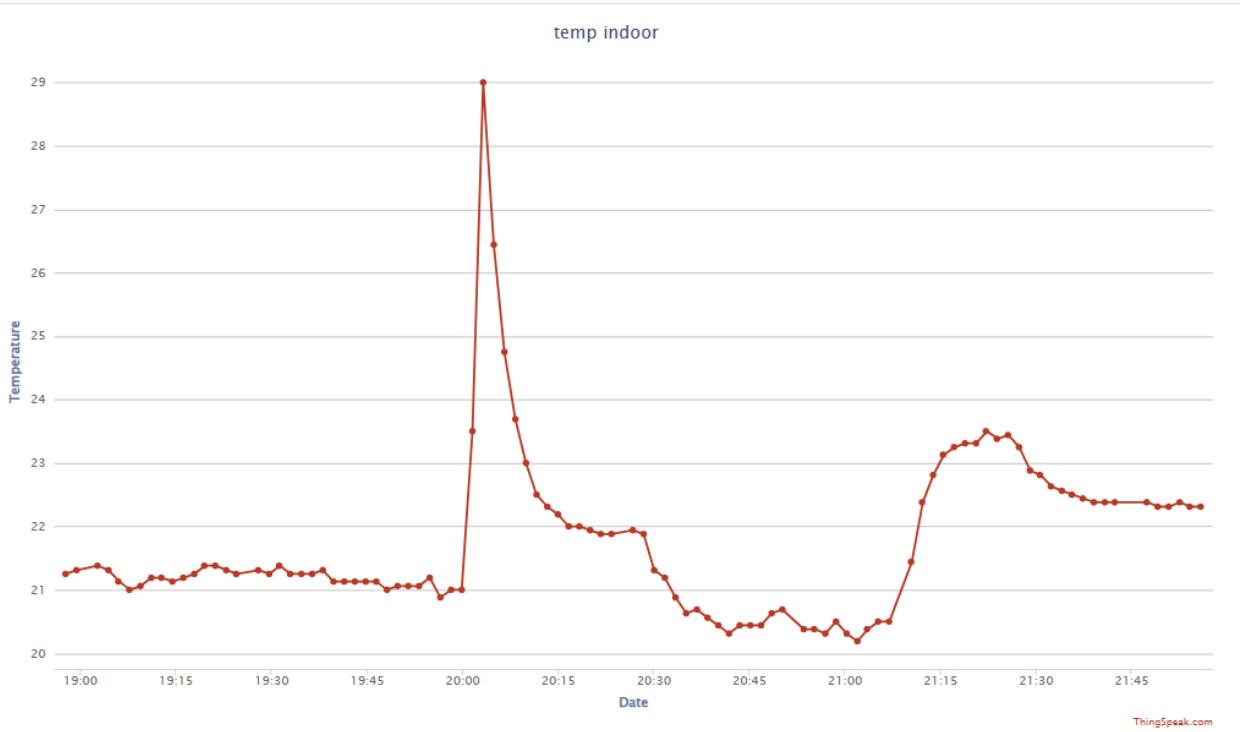

Рисунок 4.6 – Интерфейс сервиса Blynk на ПК.

По оси X шаг сетки ставится автоматически, если пользователь не захочет поставить персонализированный шаг. При наведении на точку можно узнать точную температуру и время, когда была прислано данное значение.

Далее протестируем работу системы через приложение ThingSpeak. На рисунке 4.7 приведен интерфейс приложения.

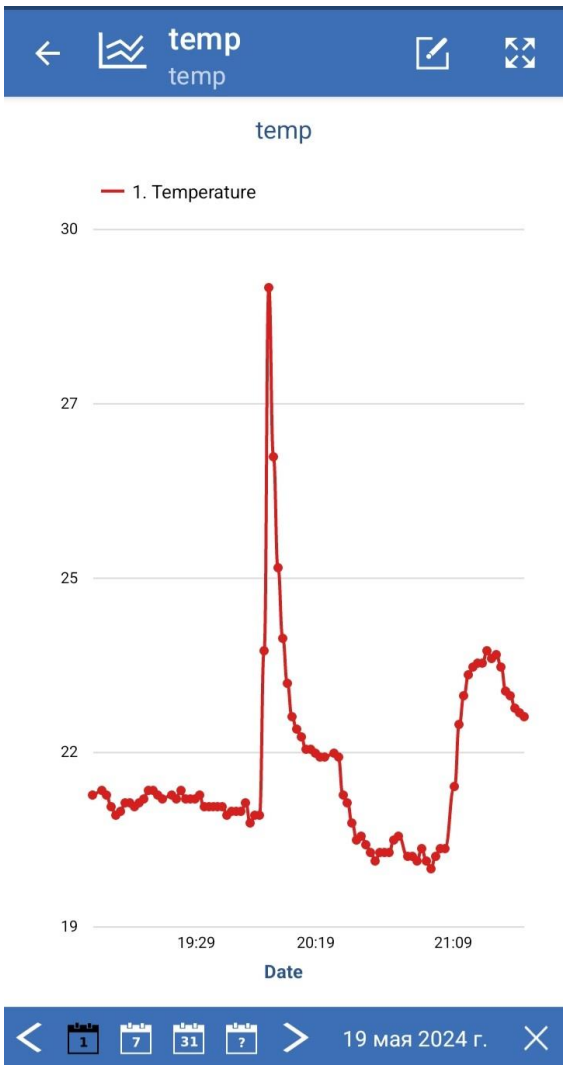

Рисунок 4.7 – Интерфейс приложения ThingSpeak.

В мобильном приложении можно масштабировать график как по горизонтали, так и по вертикали. Для этого нужно щипками расширять график по нужной оси.

Кроме того, приложение поддерживает одновременное отображение 8 графиков.

Выводы по разделу

В 4 разделе был проведен ряд экспериментов, целью которых являлась проверка корректной работы написанного кода, а также работы веб-сервиса.

5 Экспериментальные исследования программно-технического комплекса системы

Произведена сборка устройства. Аппаратный комплекс приведен на рисунке 5.1

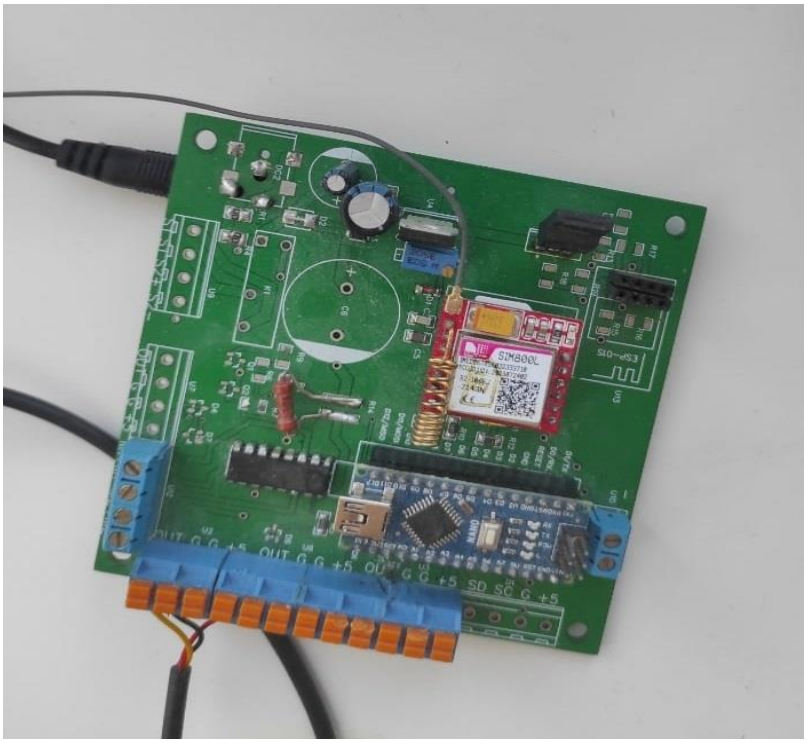

Рисунок 5.1 – Аппаратный комплекс.

Экспериментальные исследования аппаратного комплекса являются важными этапами разработки. На этом этапе выявляются все проблемные аспекты, а также подтверждается правильный режим работы устройства. В наших экспериментальных исследованиях будут следующие этапы:

— Экспериментальные исследования технической части устройства

— Экспериментальные исследования программной части устройства

— Экспериментальные исследования динамики трафика

5.1 Экспериментальные исследования технической части устройства

После успешного моделирования блока зарядки в Tina-Ti, соберем на макетной плате схему для снятия внешних характеристик, приведенную на рисунке 5.2., для того, чтобы экспериментально проверить, сможет ли спроектированный блок зарядки обеспечивать необходимую характеристику с ограничением тока 0,2 А и участком стабилизации 8,4 В.

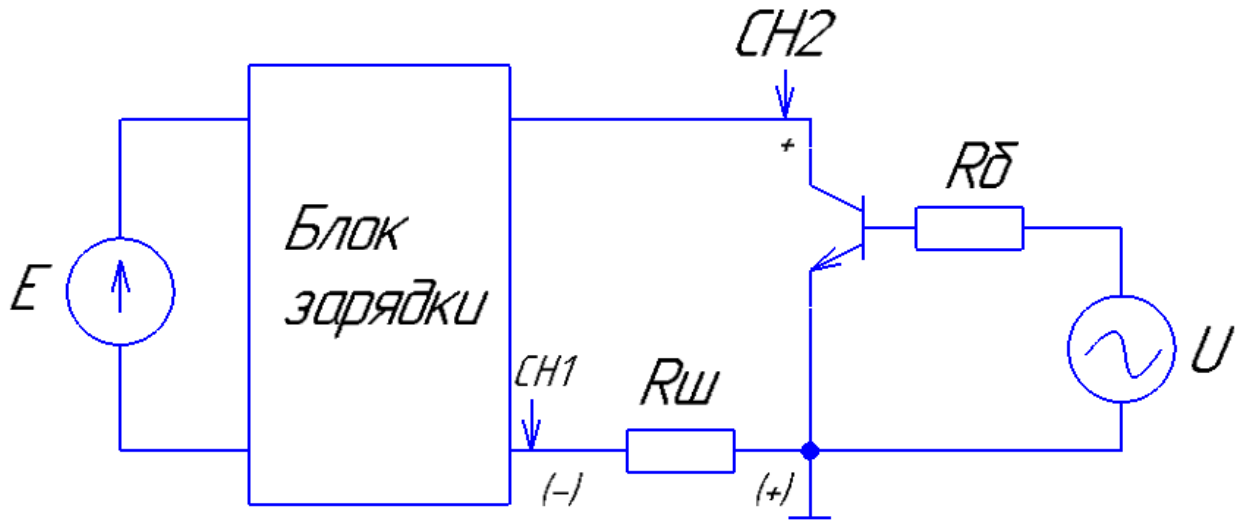

Рисунок 5.2 – Принципиальная схема для снятия внешних характеристик.

Для управления снятием внешней характеристики был выбран биполярный NPN транзистор. Это было сделано для того, чтобы облегчить сбор данных множества точек, меняя ток базы транзистора, тем самым регулировать напряжение и ток на нагрузке блока зарядки.

Схема из рисунка 5.2 состоит из:

— Rш. Это резистор, при помощи можно снимать ток на 1 канале. Для этого надо разделить напряжение, которое показывается на 1 канале на сопротивление резистора Rш. В нашем случае Rш = 0,47 Ом.

— Rб. Это резистор стабилизации рабочей точки транзистора. Он помогает контролировать ток базы и, следовательно, ток коллектора. Rб = 1 кОм.

— U. Это генератор напряжения. Он необходим для подачи на схему переменного напряжения, которое нужно для построения внешних характеристик.

На рисунке 5.3 приведены осциллограммы U(T) первого канала (тока) и второго канала (напряжение).

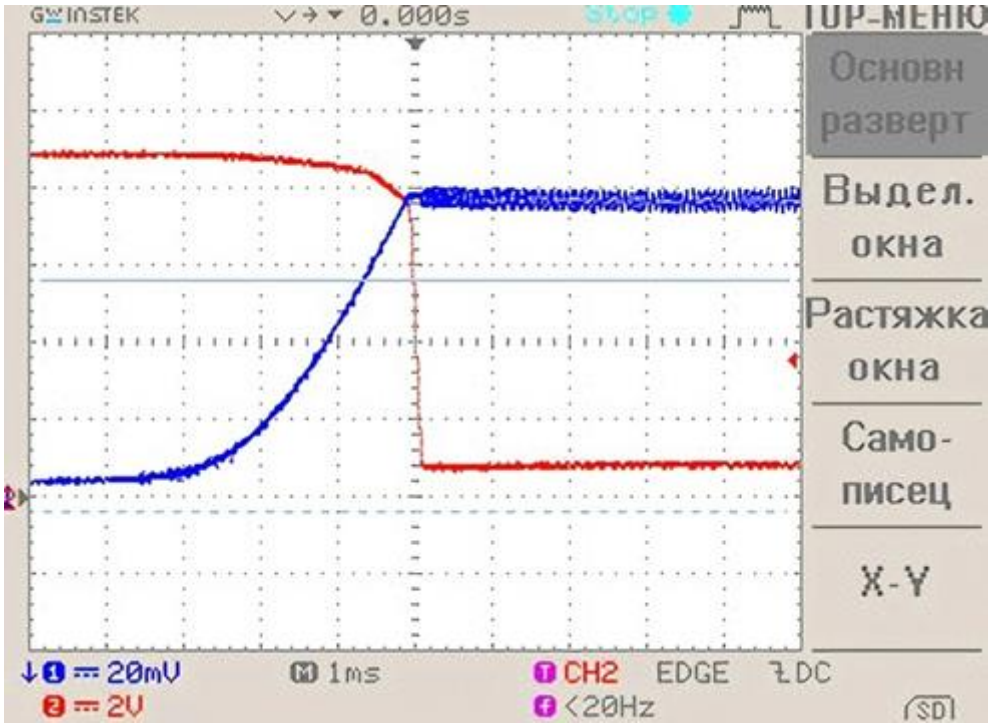

Рисунок 5.3 – Осциллограммы работы схемы при тестировании.

На рисунке 5.3 запечатлен момент, когда выходное напряжение блока зарядки падает на 2 канале, а ток возрастает. Это связано из-за того, что сопротивление остается постоянным, что подтверждает закон Ома.

На рисунке 5.4 приведена осциллограмма X/Y, где X – 1 канал (ток), Y – 2 канал (напряжение).

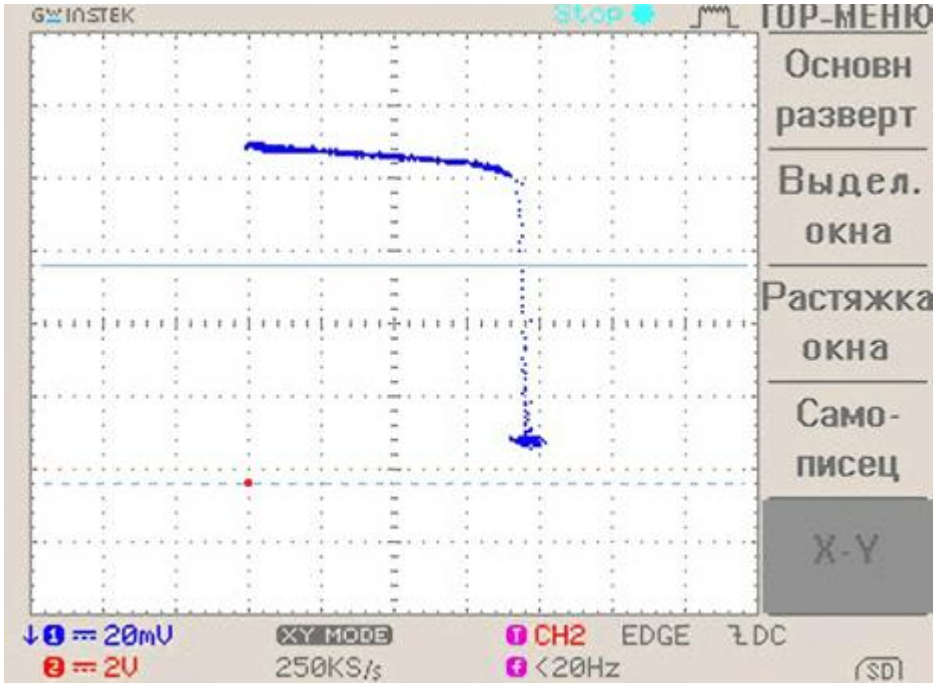

Рисунок 5.4 – Внешняя характеристика X/Y.

Красной точкой показан ноль. Как видим, на осциллограмме участок напряжения будет равен 8,4 В. Для того, чтоб узнать, какой ток ограничения имеется на внешней характеристике, воспользуется формулой 5.

$$i_{\text{цеп}} = \frac{U_{\text{цеп}}}{R} = \frac{20\ \text{мВ}}{0{,}47\ \text{Ом}} = 42{,}5\ \text{мА}, \quad (5)$$

где $U_{\text{цеп}}$-цена деления канала,В; R – сопротивление шунта, Ом.

Получается, в одной клетке по вертикали будет 42,5 мА. Начиная от точки нуля до участка стабилизации будет примерно 4,2 больших клеток. Получим, что ток ограничения равен 175 мА.

Суммируя данные выше, можем подтвердить, что блок зарядки работает как и предполагалось, выдавая напряжение 8,4 В и 190 мА.

Далее, после сборки устройства был проведен ряд экспериментов. Сначала было решено снять осциллограммы тока Arduino и модуля SIM800L в работе. Для снятия осциллограмм надо добавить две RC цепочки, на вход микроконтроллера и на общий вход. Они необходимы для фильтрации нежелательных высокочастотных или низкочастотных составляющих сигнала для нашего случая, необходимо сделать две RC цепочки, которые

подавляет высокочастотные выплески. То есть, будут работать как фильтры верхних частот.

На рисунке 5.5 приведены эти RC цепочки, это первая RC цепочка C1 с R5 и вторая цепочка R2 C6.

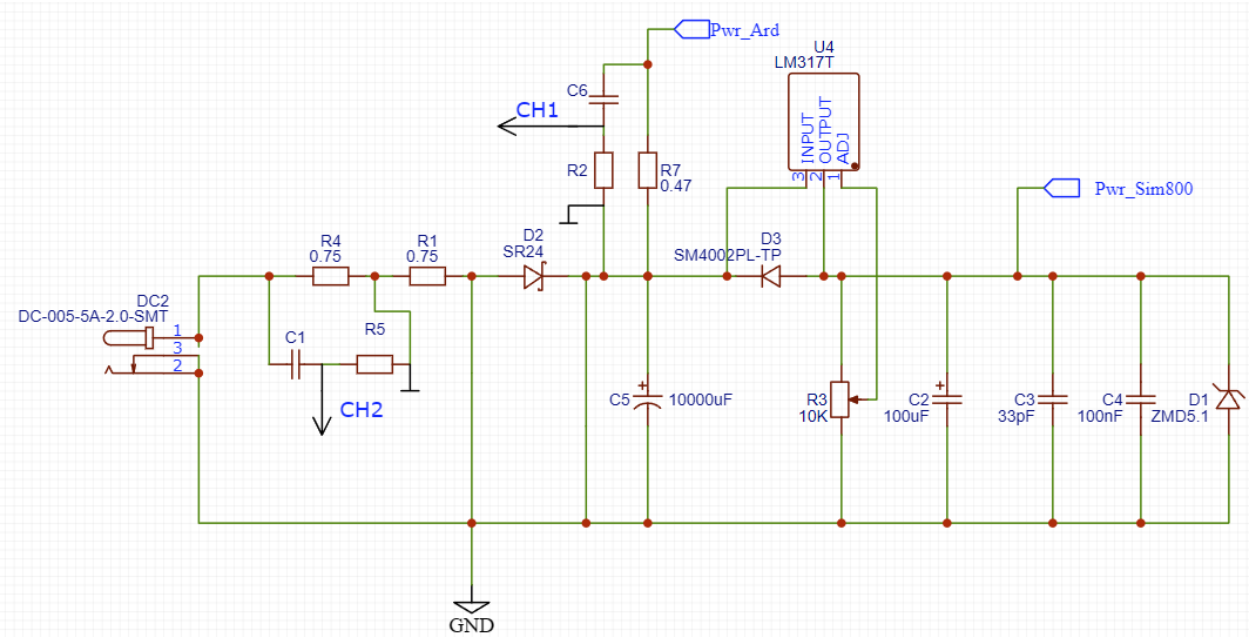

Рисунок 5.5 – Схема с RC цепочками для снятия тока Arduino и SIM800L в работе.

При помощи формул 6-7 рассчитаем значения сопротивления и ёмкости, зная частоту среза 0,2 Гц, чтобы подавить высокие частоты более 0,2 Гц.

$$f_c = \frac{1}{2 \cdot \pi \cdot R \cdot C} \quad (6)$$

$$0{,}2 = \frac{1}{6{,}28 * R * C} \quad (7)$$

Для решения уравнения, были выбраны значения C = 22 мкФ и R = 36 кОм.

Найдем постоянную времени для этой RC цепочки (формула 8).

$$\tau = R \cdot C = 22 \cdot 10^{-9} \cdot 36 \cdot 10^{4} = 0{,}8 \text{ с} \quad (8)$$

Итого, наша первая RC цепь имеет частоту среза 0,2 Гц и постоянную времени 0,8с (Постоянная времени (тау) в RC-цепи определяет, как быстро цепь реагирует на изменения во входном сигнале).

Аналогично подберем вторую RC цепь, которая будет стоять на общем входе. Были подобраны номиналы R = 770 кОм и C = 1 мкФ. Частота среза и постоянная времени такие же как и в первой RC цепи, fc=0,2 Гц и τ=0,8 с соответственно.

Кроме того, перед началом эксперимента необходимо отдельно обозначить для удобства дальнейших расчетов номиналы шунтов. Шунт, который стоит на входе к микроконтроллеру имеет сопротивление R = 0,75 Ом, на общем входе R = 0,5 Ом.

Далее, проведем ряд экспериментов со снятием осциллограмм. Для начала был снят установившийся режим (рисунок 5.6), где 1 канал – это напряжение на Arduino, а 2 канал – суммарное напряжение устройства.

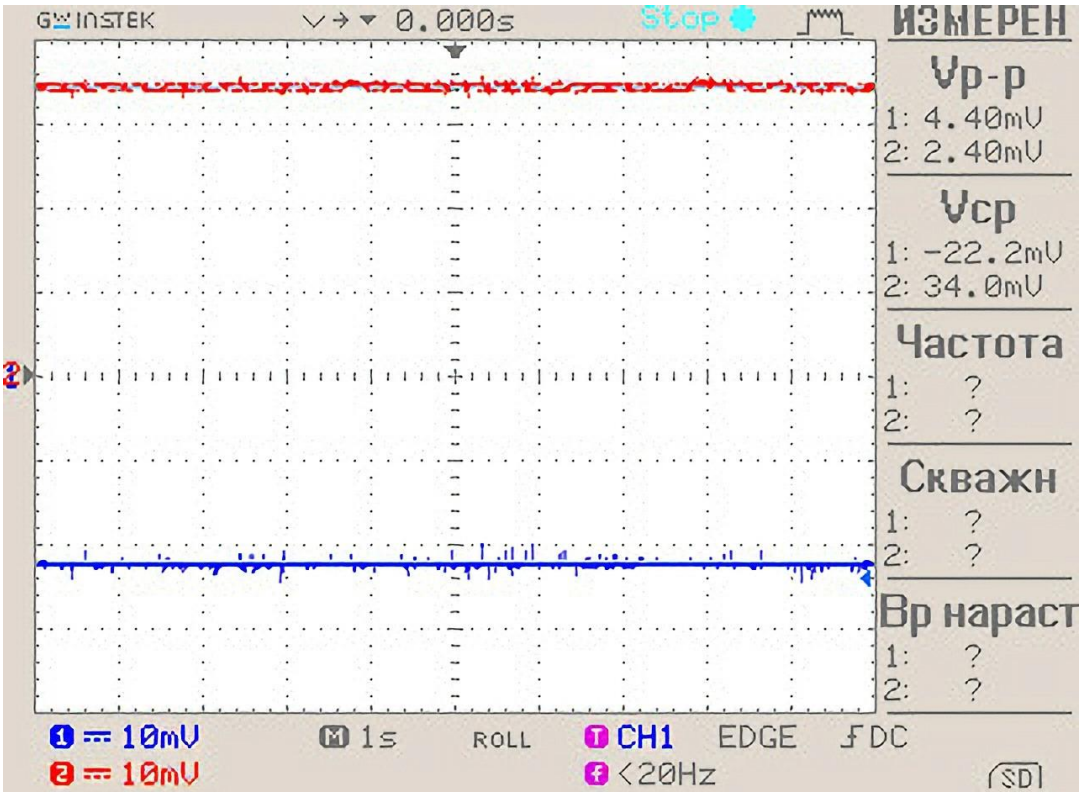

Рисунок 5.6 – Установившийся режим работы устройства.

Рассчитаем получившийся токи (формулы 9-1), где I_1- потребляемый ток на Arduino, I_2- общий ток.

$$I_1 = \frac{U_1}{R_1} = \frac{22.2*10^{-4}}{0.75} = 0{,}022 \text{ А} = 22.2 \text{ мА} \quad (9)$$

$$I_2 = \frac{U_2}{R_2} = \frac{34*10^{-4}}{0.5} = 0.068\ \text{А} = 68\ \text{мА} \quad (10)$$

Итого, чтобы узнать ток на модуле SIM800L, отнимем из общего тока ток на Arduino (формула 11). Можно так рассчитать, так как падение напряжения на остальных элементах ничтожно мало.

$$I_3 = I_2 - I_1 = 0.068 - 0{,}022 = 0{,}046\ \text{А} = 46\ \text{мА} \quad (11)$$

На основании полученных данных можно рассчитать время работы в аварийном режиме, при отключении основного источника питания (формула 12).

$$t = \frac{C}{I} = \frac{3400}{68} = 50\ \text{ч}, \quad (12)$$

где С - емкость аккумулятора, мА*ч;

I- потребляемый ток, мА.

Следующим экспериментом будет пусковая характеристика устройства (рисунок 5.7).

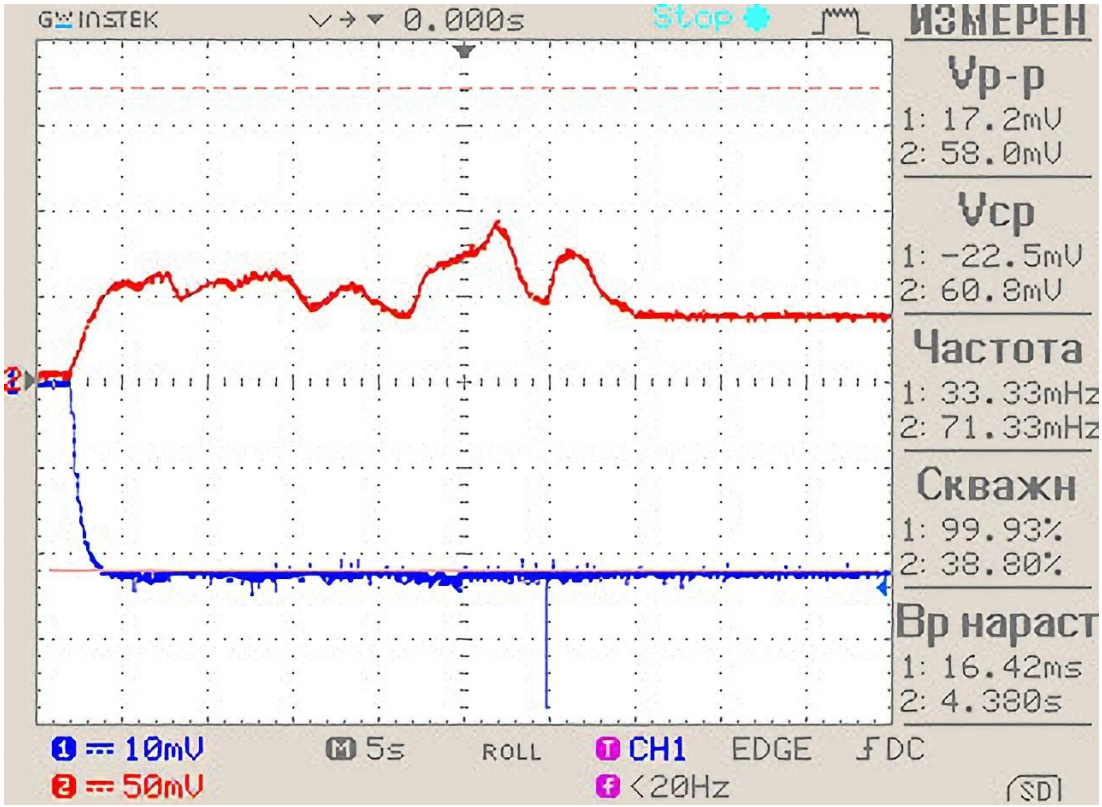

Рисунок 5.7 – Пусковая характеристика устройства.

Анализируя осциллограмму, заметим, что канал, показывающий напряжение на Arduino стабилен весь промежуток времени. Это означает, что потребление микроконтроллера более стабильно чем у SIM800L, у которой явно виден пусковой процесс по рисунку 5.7, который примерно равен половине минуты.

В это время модуль связи SIM800L находится в режиме подключения к базовой станции, поэтому имеет нестабильное напряжение. По истечении 30 секунд связь была установлена, и устройство переходит в установившийся режим работы.

Далее проведен эксперимент по выявлению, насколько сильно колеблется сигнал при пуске. Для этого поставим на 2 канал курсоры, первый – на нижнюю точку осциллограммы, второй – на вершины большинства колебаний (рисунок 5.8).

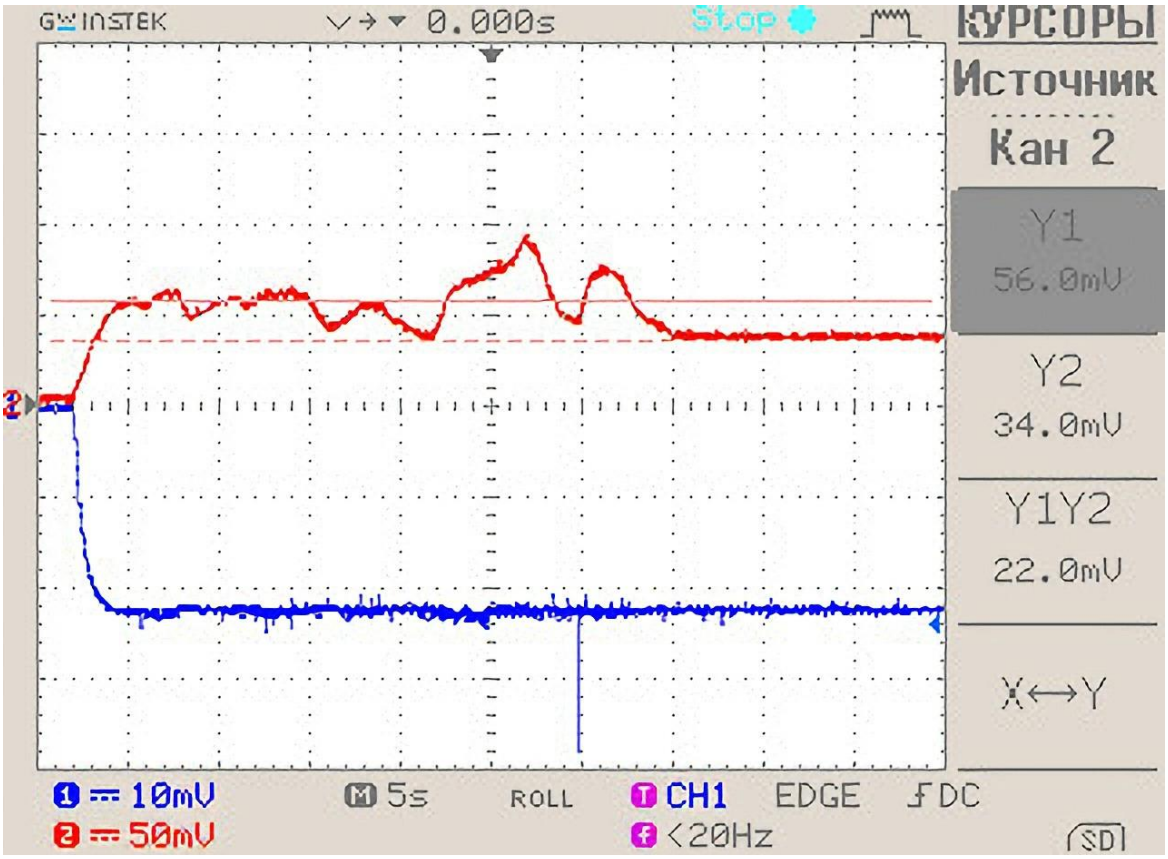

Рисунок 5.8 – Пусковая характеристика с маркерами.

Заметим, в среднем сигнал колеблется в пределах 34 мВ – 56 мВ, что является разницей всего в 1,6 раза. Так же видно что средний ток даже на пуске примерно совпадает с верхним маркером , следовательно он примерно 1,6 раза больше среднего тока в стационарном режиме.

5.2 Экспериментальные исследования программной части

В ходе тестирования были выявлены следующие недочеты:

— При одновременном попадании в один момент оповещения при помощи звонка и отправления пакетов данных, устройство иногда теряет соединение с сервисом. Происходит из-за того, что одновременно эти два действия не могут происходить. Это связано с особенностями работы микроконтроллера Arduino, который используется в данном устройстве. Arduino работает на основе однопоточного процессора, что означает, что он может выполнять только одну операцию за раз. В результате, когда устройство пытается одновременно обрабатывать звонок и отправлять пакеты данных, происходит конфликт, который может привести к потере соединения.

5.3 Экспериментальные исследования динамики трафика

В процесс ее отладки возник вопрос о расходе трафика в зависимости от числа линий датчиков температуры, чтобы точнее ответить на этот вопрос была запущена серия экспериментов. Для исследования были сделаны три эксперимента с одним, двумя и тремя датчиками температуры. Каждый датчик был настроен на передачу данных о температуре раз в две минуты. Данные собирались в течение одного дня, после чего анализировались.

Полученные данные представлены в таблице 6.

Таблица 6 – Экспериментальные данные

	1 источник		2 источника		3 источника	
час работы	Обьем	Ед изм	Обьем	Ед изм	Обьем	Ед изм
1	0,0291	Мб	0,031	Мб	0,0341	Мб
2	0,0225	Мб	0,0315	Мб	0,034	Мб
3	0,0221	Мб	0,0278	Мб	0,0346	Мб
4	0,023	Мб	0,0276	Мб	0,0337	Мб
5	0,0231	Мб	0,026	Мб	0,0471	Мб
6	0,0249	Мб	0,0286	Мб	0,0388	Мб
7	0,0232	Мб	0,0307	Мб	0,0348	Мб
8	0,0241	Мб	0,0267	Мб	0,035	Мб
9	0,0247	Мб	0,0273	Мб	0,0365	Мб
10	0,0228	Мб	0,0265	Мб	0,043	Мб
11	0,0235	Мб	0,0279	Мб	0,031	Мб
12	0,0243	Мб	0,0282	Мб	0,0352	Мб
13	0,0227	Мб	0,0273	Мб	0,0385	Мб
14	0,0243	Мб	0,0275	Мб	0,0382	Мб
15	0,0247	Мб	0,0269	Мб	0,0371	Мб
16	0,0224	Мб	0,0283	Мб	0,0366	Мб
17	0,0243	Мб	0,0262	Мб	0,0362	Мб
18	0,0219	Мб	0,0271	Мб	0,0342	Мб
19	0,0231	Мб	0,0281	Мб	0,0352	Мб
20	0,0238	Мб	0,0276	Мб	0,0361	Мб

Трафик за каждый час в зависимости от конкретного часа построен в виде графика (рисунок 5.9). Интерес представлял показатель среднего арифметического потребления трафика в каждом из случаев. Расчет этого показателя сведен в таблицу 3 и отражен на рисунке 5.9 в виде линий.

Интересно так же из данных было рассчитать расход одного пакета данных на каждую посылку показания измеренной температуры на сервер. Для этого часовой трафик нужно разделить на 30 посылок в час, так как каждый пакет отправляется раз в 2 минуты.

Далее на основе этих данных можно рассчитать расход трафика за месяц и год, что необходимо для определения по тарифным планам операторов сотовой связи стоимости обслуживания системы. Все полученные данные сведены в таблицу 7.

Немаловажным и нужным фактором в пакете каждой передачи данных на сервер является служебная информация, которая всегда постоянна и играет ключевую роль в обеспечении связи между устройствами и сервером. Эти данные несут в себе информацию о состоянии сети, аутентификации, синхронизации и другие важные параметры, которые необходимы для корректной работы системы.

Таблица 7 – рассчитанные значения расхода трафика

Объем потраченного трафика в день					
1 источник		2 источника		3 источника	
Объем	Ед. изм.	Объем	Ед. изм.	Объем	Ед. изм.
0,5716	Мб	0,6677	Мб	0,8602	Мб
Среднее значение в час					
0,023817	Мб	0,027821	Мб	0,035842	Мб
Объем одного пакета данных					
0,000794	Мб	0,000927	Мб	0,001195	Мб
Среднее значение трафика за месяц использования					
17,148	Мб	20,031	Мб	25,806	Мб
Среднее значение трафика за год использования					
205,776	Мб	240,372	Мб	309,672	Мб

Рассчитав необходимые данные, сведем их в рисунок 5.9.

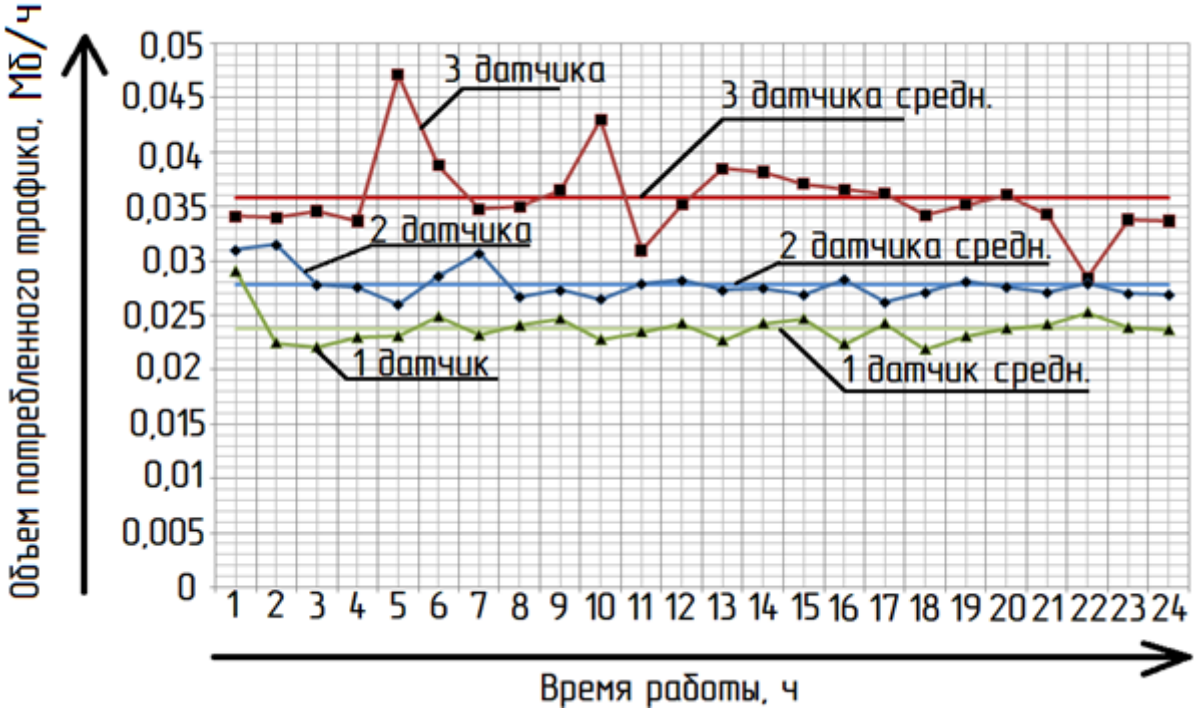

Рисунок 5.9 – График зависимости объема потребления трафика от времени работы устройства

Для вычисления служебной информации построим график зависимости объема потребленного трафика от количества датчиков (рисунок 5.10).

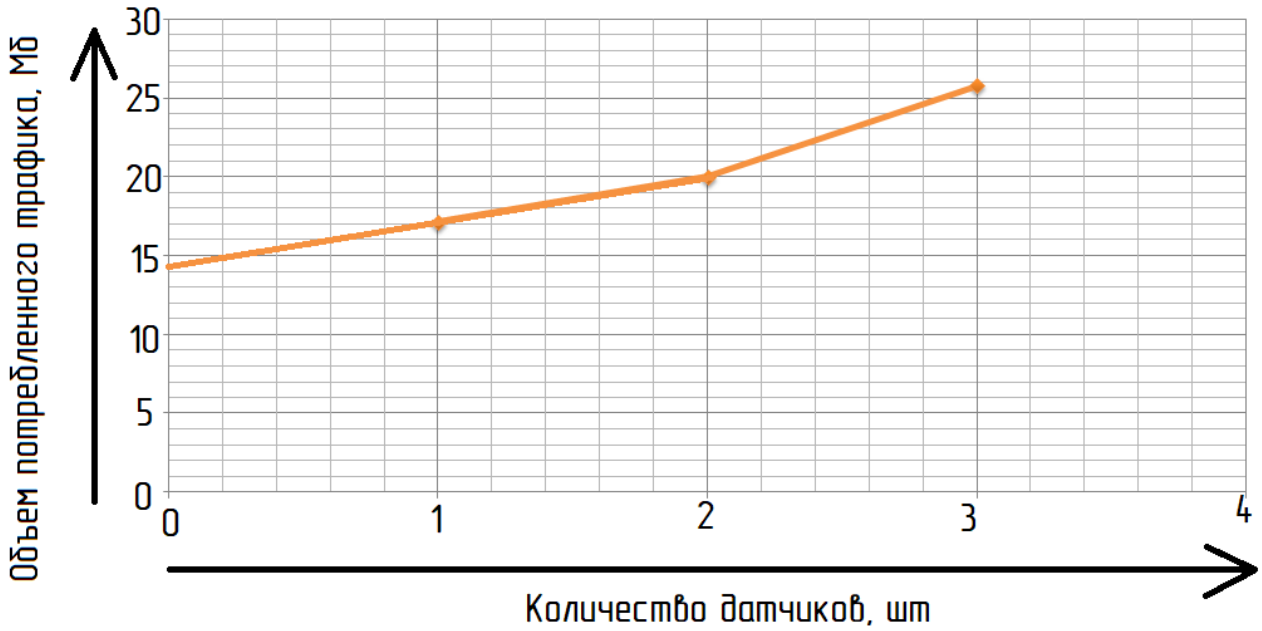

Рисунок 5.10 - График зависимости объема потребления трафика за месяц от количества датчиков

По рисунку 5.10 можно заметить, что график пересекает ось ординат в точке с нулем количеством датчиков, соответствующей объему трафика 14 Мб. Это и есть оценочно объем служебной информации, так как она отправляется независимо от наличия датчиков.

Анализируя полученные данные, можно сделать следующие выводы:

1.Объем трафика не увеличивается в два или три раза с увеличением количества источников. Связано с тем, что некоторая часть данных, такая как служебная информация, передается только один раз, независимо от количества источников. В месяц на нее расходуется примерно 14 Мб. Примерно 3 Мб трафика в месяц тратится на каждый датчик.

2. Объем трафика варьируется от часа к часу. Это может быть связано с различными факторами, такими как изменение условий окружающей среды,

нагрузка на мобильную сеть, что ведет к пересоединению модуля и генерации дополнительного служебного трафика.

3. Графики не сильно выбиваются из среднего значения, что указывает на стабильность системы в целом и в стабильную передачу данных в частности.

4. Учитывая среднемесячное потребление на три датчика 25,8 Мб, в настоящее время подойдет любой тарифный план для умных вещей от любого сотового оператора, так как он включает 100Мб трафика в месяц.

Выводы по разделу

В 5 разделе был проведен ряд экспериментов программно-технического комплекса. Была проверена работоспособность блока зарядки, а также проведен эксперимент по потреблению тока Arduino и SIM800L. Был выявлен недочет работы модуля SIM800L. Кроме того, был поставлен эксперимент по анализу потребляемого трафика устройства. Так же было рассчитано время автономной работы от АКБ.

Заключение

В ходе выполнения выпускной квалификационной работы был разработан программно-аппаратный комплекс, с помощью которого пользователь сможет взаимодействовать с домом через SMS и GSM-модуль. Эта система работает автономно, контролируя температуру и электроприборы, а также сообщает пользователю о текущем состоянии и возможных проблемах. Комплекс сможет отправлять данные о температуре в доме через интернет на специальный веб-сервер.

Пользователь сможет просматривать графики и статистику температур за разные периоды через сайт, либо же посредством приложения на android, что поможет анализировать эффективность системы отопления и оптимизировать её работу. Также может при помощи отправки SMS узнать текущую температуру. Кроме того, у системы имеется «белый список» и работает только с верифицированными пользователями. Это предотвращает попытки злоумышленников управлять системой, посылая команды с неизвестных или поддельных номеров.

Разработанный программно-аппаратный комплекс может работать автономно 50 часов. Объем трафика варьируется от часа к часу, но в среднем тратит устройство в месяц не более 30 Мб интернет трафика при подключении 3 датчиков температуры.

К устройству можно подключить любую нагрузку, мощность которой не превышает 2 кВт.

Простота интерфейса делает комплекс доступным для широкого круга пользователей, не требующий специальных знаний в области электроники.

Таким образом, созданный программно-аппаратный комплекс является надежным и эффективным решением для управления домом, обеспечивая пользователю удобство и безопасность при минимальных эксплуатационных затратах.

Список используемых источников

1. Блум Д. Изучаем Arduino. Инструменты и методы технического волшебства. СПБ.:BHV, 2020. 544 с.

2. Источник опорного напряжения TL431AA 36V 0.5% TO-92 [Электронный ресурс]. URL: https://asenergi.ru/catalog/microchip/power/tl431aa-to-92.html (дата обращения: 3.05.2024).

3. Интернет для вещей [Электронный ресурс]. URL: https://msk.tele2.ru/tariff/iot (дата обращения: 3.05.2024).

4. Накопительный электрический водонагреватель EDISSON ER 100 V ЭдЭ001798 [Электронный ресурс]. URL: https://www.vseinstrumenti.ru/product/nakopitelnyj-elektricheskij-vodonagrevatel-edisson-er-100-v-ede001798-1570441/ (дата обращения: 04.04.2024).

5. Предельная рассеивая мощность TO220 [Электронный ресурс]. URL: https://u.to/nWy2IA (дата обращения: 04.02.2024).

6. Тариф «Для устройств» — специальное решение для умных вещей [Электронный ресурс]. URL: https://media.mts.ru/internet/186434/ (дата обращения: 5.03.2024).

7. Телематика для M2M/IoT-устройств [Электронный ресурс]. URL: https://moskva.beeline.ru/business/iot/telematika-m2m/ (дата обращения: 3.05.2024).

8. Типы аккумуляторов: обзор и сравнение [Электронный ресурс]. URL: https://silverkomp.ru/poleznoe/tipy-akkumulyatorov-obzor-i-sravnenie (дата обращения: 18.04.2024).

9. Умные вещи [Электронный ресурс]. URL: https://moscow.megafon.ru/tariffs/all/umnye_veschi.html (дата обращения: 1.02.2024).

10. Цифровой датчик температуры DS18B20 [Электронный ресурс] URL : https://3d-diy.ru/wiki/arduino-datchiki/tsifrovoy-datchik-temperatury-ds18b20/ (дата обращения 30.07.2023).

11. Что такое интернет вещей и где применяются IoT устройства [Электронный ресурс]. URL: https://auroraevernet.ru/articles/chto-takoe-internet-veshchey-i-gde-primenyayutsya-iot-ustroystva/ (дата обращения: 3.04.2024).

12. Bms 2s схема подключения зарядки [Электронный ресурс]. URL: https://chistotnik.ru/bms-2s-skhema-podklyucheniya-zaryadki.html (дата обращения: 18.04.2024).

13.Arduino Nano [Электронный ресурс]. URL: https://arduino.ru/Hardware/ArduinoBoardNano (дата обращения: 06.02.2024).

14. Arduino Nano [Электронный ресурс]. URL: https://kit.alexgyver.ru/tutorials/arduino-nano/ (дата обращения: 3.04.2024).

15. EasyEDA [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/EasyEDA (дата обращения: 07.05.2024).

16. GSM термометр ALONIO T2 [Электронный ресурс] URL: https://alonio.ru/catalog/alonio-t2/ (дата обращения: 06.02.2024).

17 GSM датчик температуры «Полюс GSM Термо» [Электронный ресурс]. URL: https://arsenal-sib.ru/gsm_signalizacija/avtonomnaya/polus_thermo/ (дата обращения: 06.02.2024).

18. HS 201-30, Радиатор 30x23x16 мм, 19 дюйм*градус/Вт [Электронный ресурс]. URL: https://www.chipdip.ru/product/hs-201-30 (дата обращения: 3.05.2024).

19. LM317 [Электронный ресурс]. URL: https://www.ti.com/lit/ds/symlink/lm317.pdf (дата обращения: 06.02.2024).

20. SRD-05VDC-SL-C [Электронный ресурс]. URL: https://chipdocs.ru/articles/srd-05vdc-sl-c/ (дата обращения 30.07.2023).

21. SIM800L GSM Module [Электронный ресурс]. URL: https://components101.com/wireless/sim800l-gsm-module-pinout-datasheet-equivalent-circuit-specs (дата обращения: 06.02.2024).

22. TL431: схема, характеристики, datasheet и аналоги [Электронный ресурс]. URL: https://remontka.com/tl431/ (дата обращения: 3.05.2024).

23. Temperature monitoring RTU5023 [Электронный ресурс]. URL: https://www.iot-solution.com/temperature-monitoring-p00199p1.html (дата обращения: 06.02.2024).

Приложение А

Программный код для Arduino Nano

```
#include <SoftwareSerial.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 3
#define RELAY_PIN 4
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
SoftwareSerial gprsSerial(6,7);
float tempC;
String whiteList[] = {"+79060000000"};
int numNumbers = sizeof(whiteList) / sizeof(whiteList[0]);

void sendSMS(String number, String message) {
gprsSerial.print("AT+CMGF=1\r");
delay(1000);
gprsSerial.print("AT+CMGS=\"" + number + "\"\r");
delay(1000);
gprsSerial.print(message);
delay(1000);
gprsSerial.print((char)26);
delay(1000);
gprsSerial.println();
Serial.println("Сообщение отправлено");
delay(1000);
}
void callNumber(String number) {
gprsSerial.println("ATD" + number + ";");
Serial.println("Аварийный звонок совершен");
delay(30000);
gprsSerial.println("ATH");
}

void checkIncomingCall(float tempC) {
while (gprsSerial.available()) {
String incomingData = gprsSerial.readString();
if (incomingData.indexOf("+CLIP:") != -1) {
for (int i = 0; i < numNumbers; i++) {
if (incomingData.indexOf(whiteList[i]) != -1) {
Serial.println("Звонок от номера из белого списка");
sendSMS(whiteList[i], "Actual temp" + String(tempC));
delay(3000);
}
}
}
}
}
void setup()
{
  gprsSerial.begin(9600);
```

Продолжение приложения А

```
Serial.begin(9600);
sensors.begin();
delay(1000);
gprsSerial.println("AT"); // Отправка команды AT
 delay(1000);
 if (gprsSerial.find("OK")) {
 Serial.println("Команда AT выполнена успешно");
  } else {
   Serial.println("Ошибка команды AT");
  }
delay(1000);
  gprsSerial.println("AT+CLIP=1"); // Включение CLIP
  delay(1000);
  if (gprsSerial.find("OK")) {
    Serial.println("Команда AT+CLIP=1 выполнена успешно");
  } else {
    Serial.println("Ошибка команды AT+CLIP=1");
  }
}
void loop()
{
sensors.requestTemperatures();
float tempC = sensors.getTempCByIndex(0);
Serial.print("Текущая температура: ");
Serial.println(tempC);
checkIncomingCall(tempC);
gprsSerial.println("AT");
delay(1000);
gprsSerial.println("AT+CPIN?");
delay(1000);
gprsSerial.println("AT+CREG?");
delay(1000);
gprsSerial.println("AT+CGATT?");
delay(1000);
gprsSerial.println("AT+CIPSHUT");
delay(1000);
gprsSerial.println("AT+CIPSTATUS");
delay(2000);
gprsSerial.println("AT+CIPMUX=0");
delay(2000);
gprsSerial.println("AT+CSTT=\"airtelgprs.com\"");
delay(1000);
gprsSerial.println("AT+CIICR");//bring up wireless connection
delay(3000);
gprsSerial.println("AT+CIFSR");//get local IP adress
delay(2000);
gprsSerial.println("AT+CIPSPRT=0");
delay(3000);
gprsSerial.println("AT+CIPSTART=\"TCP\",\"api.thingspeak.com\",\"80\"");
delay(6000);
```

Продолжение приложения А

```
gprsSerial.println("AT+CIPSEND");
delay(4000);
String str = "GET https://api.thingspeak.com/update?api_key=VVS7LH6YT75J3YOO&field1=0" + String(tempC);
Serial.println(str);
gprsSerial.println(str);
delay(4000);
gprsSerial.println((char)26);
delay(5000);
gprsSerial.println();
gprsSerial.println("AT+CIPSHUT");
delay(300);
checkIncomingCall(tempC);
delay(60000);
if (tempC < 20.0) {
for (int i = 0; i < numNumbers; i++) {
sendSMS(whiteList[i], "Warning: Temperature is below 20 degrees Celsius!");
delay(3000); // Ждем минуту
callNumber(whiteList[i]);
```