Общество с ограниченной ответственностью

"ИНСАЙТ-менеджмент"

УДК 004.78, 004.41

Инвентарный №

УТВЕРЖДАЮ:

Генеральный директор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.К. Грибанова

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2017 г.

О Т Ч Е Т

О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ

И ОПЫТНО-КОНСТРУКТОРСКОЙ РАБОТЕ (НИОКР)

Разработка программного обеспечения микроконтроллера модуля счетно-импульсного ввода

(промежуточный)

Этап № 3

Ген. директор

ООО «ИНСАЙТ-Электроникс» С.В. Грибанов

Руководитель НИОКР, к.т.н. А.В. Кычкин

Исполнитель С.А. Артемов

Пермь 2018СПИСОК ИСПОЛНИТЕЛЕЙ

Руководитель работы,

к.т.н. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Кычкин

подпись, дата

Исполнители темы:

Артемов С.А.

Реферат

Отчет 21 с., 2 рис.

**Цель работы –** целью работы является формирование общей структуры проекта, описание недостающих узлов и объектов системы. Таких как: Python фреймворк, api уровень верхнего порядка, загрузчик, прошивка проекта WI-FI связи. А так же тестирование сетевого интерфейса Ethernet на устойчивость.СОДЕРЖАНИЕ

[Реферат 3](#_Toc504145633)

[Перечень условных обозначений и сокращений 5](#_Toc504145634)

[ВВЕДЕНИЕ 6](#_Toc504145635)

[ОСНОВНАЯ ЧАСТЬ 7](#_Toc504145636)

[1 Структурная схема проекта трехфазного регистратора 7](#_Toc504145637)

[2 Описание Python Framework 9](#_Toc504145638)

[2.1 Описание разработки библиотечных модулей 11](#_Toc504145639)

[2.3 Пример использования стандартных функций 13](#_Toc504145640)

[3 Описание модуля ESP8266 14](#_Toc504145641)

[4 Описание модуля загрузчика 15](#_Toc504145642)

[5 Тестирование стабильности работы Ethernet соединения 17](#_Toc504145643)

[ЗАКЛЮЧЕНИЕ 20](#_Toc504145644)

Перечень условных обозначений и сокращений

CRC - Cyclic redundancy check. Циклический избыточный код

SoC – System on Chip. Система на кристалле

IGMP **-** Internet Group Management Protocol. Протокол управления группами Интернета.

LwIP - lightweight IP. TCP/IP-стек с открытым исходным кодом

OSI - open systems interconnection. Модель Взаимодействия Открытых Систем.

RTC – Real time clock. Часы реального времени.

SNTP - Simple Network Time Protocol. протокол синхронизации времени.

UART - Universal Asynchronous Receiver-Transmitter. Универсальный асинхронный приёмопередатчик.

ВВЕДЕНИЕ

Основным направлением работы данного этапа является формирование структуры проекта, его описание и дополнение ранее не рассмотренными функциональными блоками. Для этого, составлена общая структура системы, описано содержимое блоков, их функции и взаимосвязи.

Далее описана реализация ранее не реализованных блоков, их функционал и особенности реализации.

Основным узким местом системы является связь, так как основным каналом связи является Ethernet, его необходимо проверить в текущей реализации на устойчивость канала связи. Таким образом, произведено тестирование канала связи на физическом и транспортном уровне модели OSI.

ОСНОВНАЯ ЧАСТЬ

1 Структурная схема проекта трехфазного регистратора

На рисунке 1 приведена структурная схема проекта. На ней отображены основные элементы и взаимосвязи библиотек и функциональных элементов.

В проекте два канала связи с сервером, в каждом канале реализованы протоколы связи TCP/UDP и SNTP. Для работы с устройством Ethernet используется наиболее простой и функциональный стек LwIP, который имеет ряд преимуществ, приведенных в отчете номер 2. Реализация идентичных протоколов связи для канала WI-FI выведена в отдельный файл прошивки для микросхемы ESP8622. Для работы с данной микросхемой применяется интерфейс UART.

Выбор канала связи производится селектором потока, селектор выбирает поток на основе действующих интерфейсов и текущего источника пакета.

Работа с SNTP производится непосредственно драйверами устройств связи и напрямую обновляет время RTC.

В зависимости от номера сокета пакет TCP/UDP поступает на MQTT обработчик или обработчик удаленного управления.

Обработчик удаленного управления реализует протокол отладки, управления интерфейсам и Python машиной напрямую.

MQTT обработчик реализует протокол и формирует подписки к топикам по заранее настроенным именам. В канале топика MQTT сформирована структура JSON сообщений, для упаковки и парсинга сообщений применяется JSON библиотека.

Для работы с картой памяти реализована библиотека FatFS, она позволяет работать с картой памяти по интерфейсу SDIO.

Интерфейс USB включает два профиля подключения. Первый профиль – накопитель, который взаимодействует с библиотекой FatFS и формирует канал между картой памяти и драйвером компьютера. Второй профиль VCP (виртуальный последовательный порт). Данный профиль формирует тоннель в USB интерфейсе и реализует последовательный порт для подключения консольных приложений или приложений отладки. Для работы с устройством через VCP разработан проприетарный протокол отладки и управления.

Flash память устройства разбита на блоки кратные страницам памяти. В памяти ранятся логи работы устройства, настройки параметры, программы Python скрипта в бинарном виде и прошивка загрузчика.

Загрузчик устройства предназначен для обновления прошивки устройства через каналы связи USB или TCP/IP подключение. Для выполнения обновления необходимо закачать бинарный файл прошивки в область памяти внешней Flash памяти, настроить параметры обновления и перезапустить устройства, после чего загрузчик обновит программу перейдет по адресу начала прошивки.

Основной большей частью прошивки является Python Framework, который реализует свободно программируемую часть устройства. Для ее работы необходим скрипт в бинарном виде. Для написания скрипта используется любой редактор Python кода. Далее данный код преобразуется в бинарный с помощью специального конвертера и на выходе получаем код, который необходимо записать во Flash память через TCP/IP подключение, USB интерфейс или SD карту памяти.

Для взаимодействия фрейморка с внешней периферией необходимы библиотеки. Данный библиотеки состоят из двух частей: первая нативные функции на языке программирования проекта, скелет данных функций формирует так же генератор бинарных скриптов и непосредственно библиотечные модули самого Python.

Для работы с модулями их необходимо в тексте скрипта подключить в первой строке и работать лишь с api функционалом верхнего уровня.

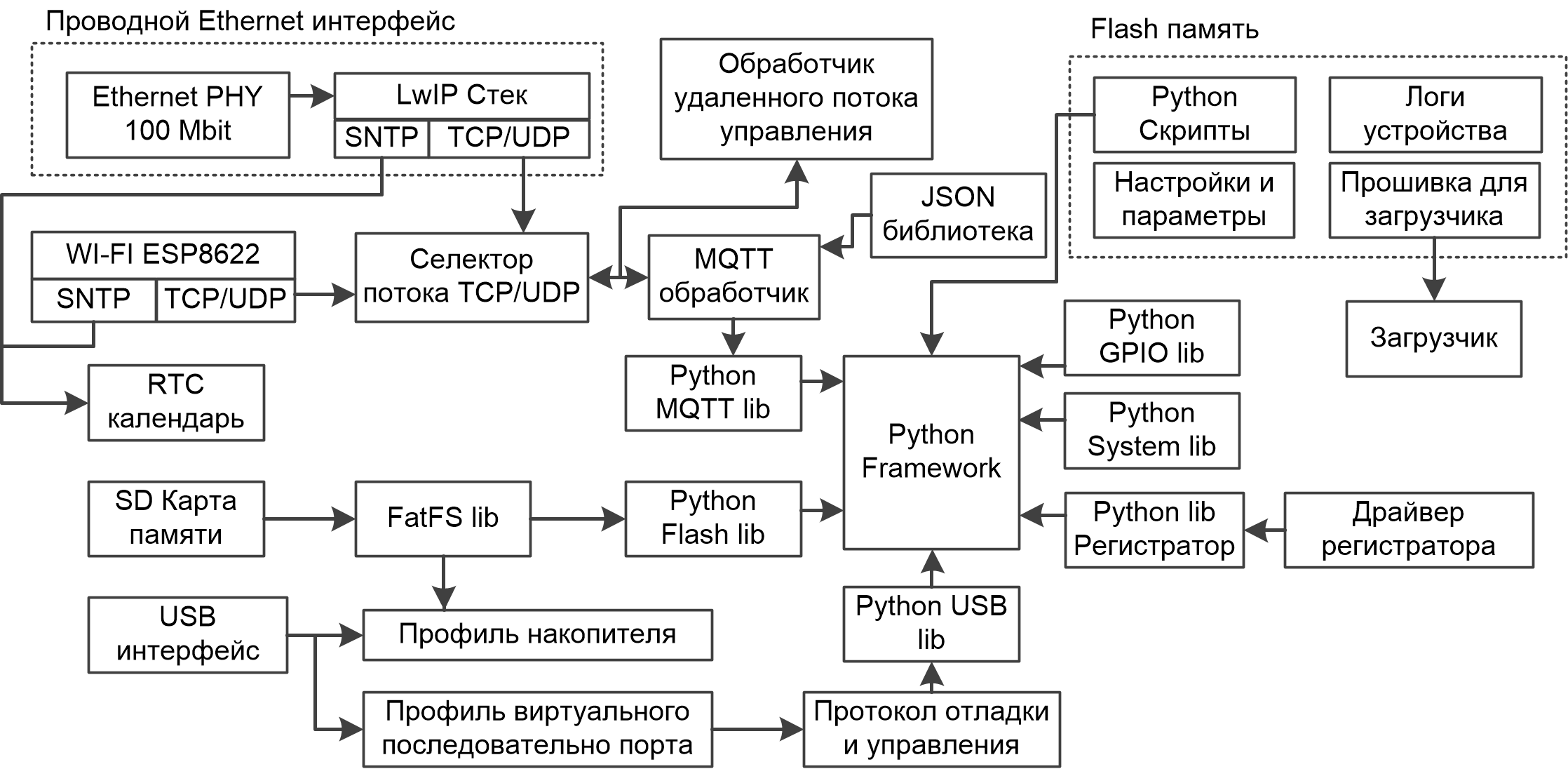


Рисунок 1 – структурная схема проекта трехфазного регистратора

Некоторые модули уже были описаны в предыдущих отчетах, далее будут описаны остальные модули.

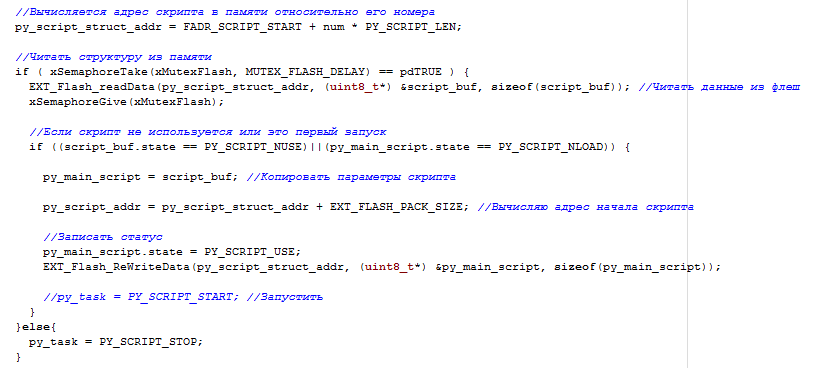
2 Описание Python Framework

Python фреймворк основан на проекте PyMite. Данный проект рассчитан на работу с версией языка Python 2.5 и предназначен для контроллеров с минимальным размерами памяти. В исполнении разработчиков фреймворка, данный проект работает в бесконечном основном цикле программы контроллера.

В случае данного проекта используется операционная система FreeRTOS, которая позволяет выделить отдельную задачу для фреймворка и путем изменения приоритета задачи добиться наибольшей производительности в мультипоточном исполнении проекта.

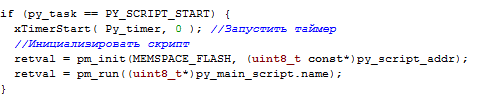
PyMite так же имеет возможности мультипоточности скриптов. Например, можно запустить скрипты для отдельных не зависящих задач, таких как аварийный контроль и основной скрипт регистратора. Однако процессорное время так же будет разделено согласно загрузке скриптов.

Для работы фреймворка необходимо загрузить скрипт по адресу в памяти. В качестве памяти хранения скрипта используется внешняя flash память. Память содержит набор скриптов под уникальным номером. Каждый скрипт имеет свой заголовок в памяти, где хранится имя и его размер. (Листинг 2.1)



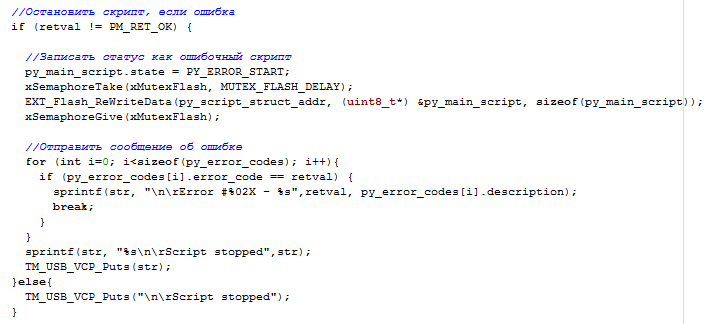
Листинг 2.1 – загрузка скрипта

Для запуска скрипта используется следующая структура (листинг 2.2). Вычисляется адрес в памяти, указывается тип памяти и запускается скрипт.



Листинг 2.2 – запуск скрипта

Если по какой-либо причине возникла ошибка, вызывается исключение и программа в цикле выполняется далее, где выводится сообщение об ошибке (Листинг 2.3).



Листинг 2.3 – обработка ошибки скрипта

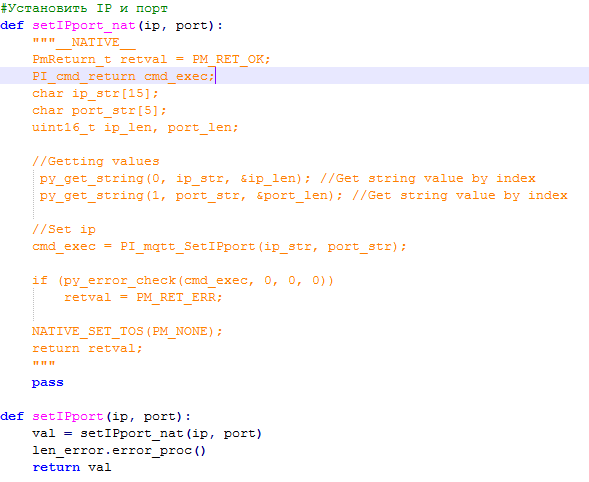
2.1 Описание разработки библиотечных модулей

Библиотечные модули фреймворка делятся на два вида – это стандартные и пользовательские. Пользовательские функции представлены в виде бинарного кода и относятся к скрипту непосредственно, недостаток их реализации – это большой объем выделяемой памяти и низкая скорость отработки, таким образом, необходимо воздерживаться от их использования и в большинстве случаев вывести функции в стандартные библиотечные модули.

Стандартные генерируются с помощью данной конструкции Листинг 2.4.

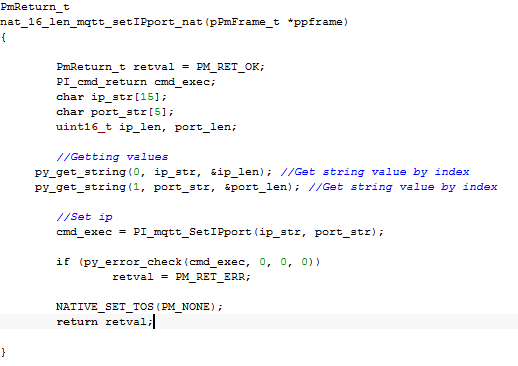
Функция setIPport состоит из одной подфункции с индексом \_nat и строки проверки на ошибку.

В подфункции setIPport\_nat в области \_\_NATIVE\_\_ записывается код на языке c, который и будет в последствии исполняться сгенерированной функцией.



Листинг 2.4 – стандартная функция в коде Python

Для работы данной библиотечной стандартной функции с помощью генератора, получаем файлы со списоком нативных функций (листинг 2.5) и файл с бинарным кодом каждой функции, объединенные в один бинарный массив.



Листинг 2.5 – представление функции в компиляторе IAR

2.3 Пример использования стандартных функций

В качестве примера возьмем вывод сигнала ШИМ с изменяемым значением по интервалу времени (листинг 2.6).

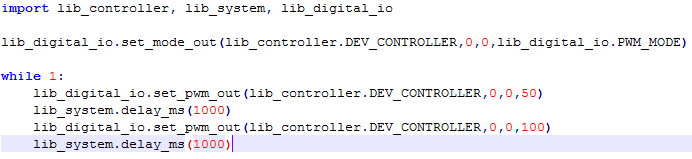
Для этого необходимо в любом проекте добавить модуль lib\_controller, который портирует функционал контроллера.

Для работы с ситемными функциями, такими как задержка используется системная библиотека lib\_system.

Для работы с ШИМ непосредственно используется библиотека lib\_digital\_io.

Для начала данные библиотеки указываются в разделе import.

Далее производится настройка режима вывода сигнала и далее переключение параметров ШИМ сигнала.



Листинг 2.6 – пример вывода ШИМ

Таким образом, получаем api уровень функций, который будет отрабатываться на конкретном устройстве.

Для работы в режиме мультипоточности необходимо добавить еще один цикл while в скрипт, таким образом, получаем два цикла, которые не зависят друг от друга. Однако стоит проверить их отдельно заранее, поскольку зависание и исключение приведет к неработоспособности фреймворка в целом, что затронет и другие задачи.

3 Описание модуля ESP8266

Для работы с сетями Wi-Fi используется SoC контроллер ESP8266, который имеет внешнюю Flash память для хранения прошивки.

В качестве стандартной прошивки производитель предлагает использовать прошивку NodeMCU, которая имеет набор модулей, включая MQTT и SNTP, однако набором AT команд, по которым будет осуществляться связь с модулем, имеет ограниченный функционал, для этого необходимо дополнять и дописывать данную прошивку.

Еще одним решением прошивки модуля является собственный алгоритм, который пишется на языке С++. Таким образом была написана прошивка для модуля ESP8266 в которой реализован канал MQTT.

AT команды представлены далее:

**Настройки подключения**

AT+MQTT\_SERVER = “Имя MQTT сервера”

AT+MQTT\_SERVER? //Получить имя сервера

AT+MQTT\_PORT=”Номер порта подключения”

AT+MQTT\_PORT? //Получить порт подключения

AT+MQTT\_USER =”Имя пользователя для защищенного подключения”

AT+MQTT\_USER? //Получить имя пользователя

AT+MQTT\_PASS=”Пароль для защищенного подключения”

AT+MQTT\_PASS? //Получить пароль

**Настройки подписок**

AT+TOPIC\_SUBCRIBE=”Имя топика для подписки”

AT+TOPIC\_SUBCRIBE?

AT+TOPIC\_PUBLISH=”Имя топика для публикации сообщений”

AT+TOPIC\_PUBLISH?

**Общие настройки**

AT+WIFI\_SSID=”Имя сети”

AT+WIFI\_SSID?

AT+WIFI\_PASS=”Пароль сети”

AT+WIFI\_PASS?

**Работа с сообщениями**

AT+SEND=”Содержимое для публикации”

AT\_MESSAGE=”Полученное сообщение”

**Команды**

AT+STATE& //Состояние модуля (OK)

AT+VERSION& //Версия прошивки

AT+RESET& //Сброс модуля

AT+DEFAULT& //Установка настроек по умолчанию

AT+WIFI\_CONN? // 1- WIFI подключен, 0-подключения нет

AT+MQTT\_CONN? // 1-MQTT подключен, 0-подключения нет

Для работы с модулем сначала нужно поменять дефолтные настройки имени сети, пароля, назначить IP адрес MQTT брокера и задать логин и пароль брокера. Далее подключиться к необходимому топику брокера. После прихода сообщения в терминале появиться сообщение AT\_MESSAGE, где и будет содержаться JSON структура сообщения. Для отправки необходимо применить функцию AT+SEND.

Данная прошивка протестирована на предыдущих устройствах, текущая версия 2.1.

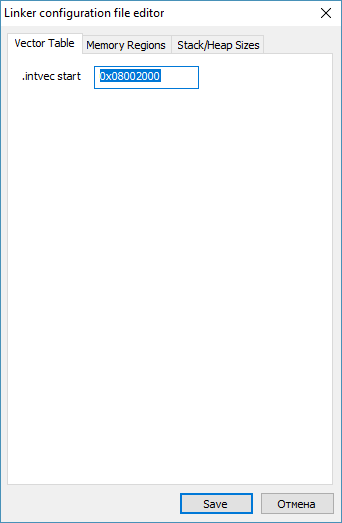
4 Описание модуля загрузчика

Модуль загрузчика позволяет перепрограммировать устройство удаленно или локально без использования программатора.

Загрузчик является отдельным проектом в среде IAR, таким образом, он не зависит от основного ПО и расположен в фиксированном адресном пространстве. Адресом начала загрузчика является начало Flash памяти контроллера, в этом случае контроллер начнет выполнение загрузчика.

Начало самого основного ПО является конец загрузчика, при учете привязки сектора памяти.

Для этого необходимо в проекте основной прошивки в настройке линкера установить адрес вектора и адрес начала flash памяти (рисунок 4.1).



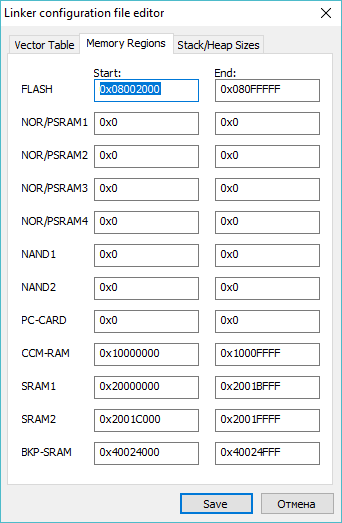
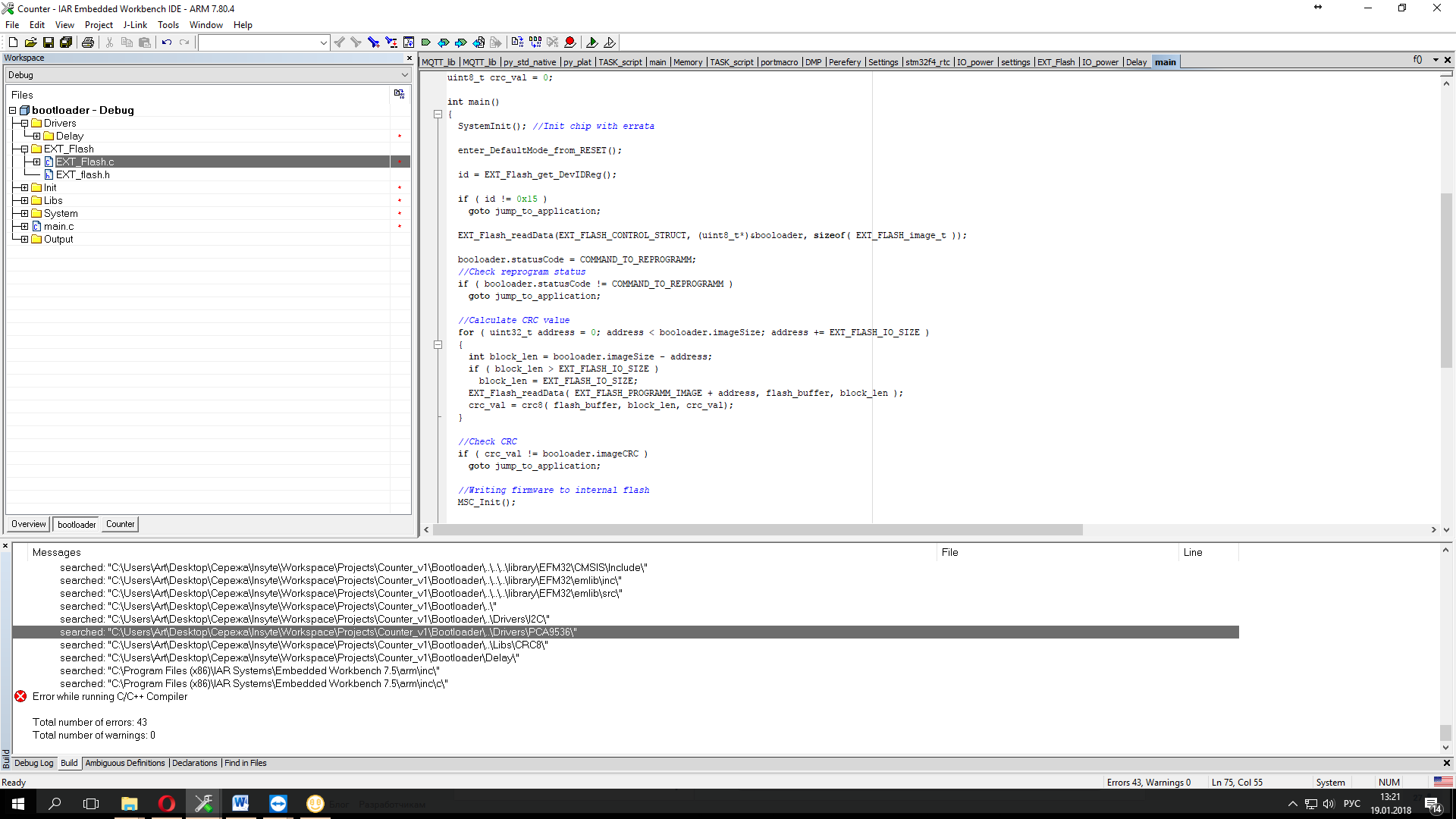


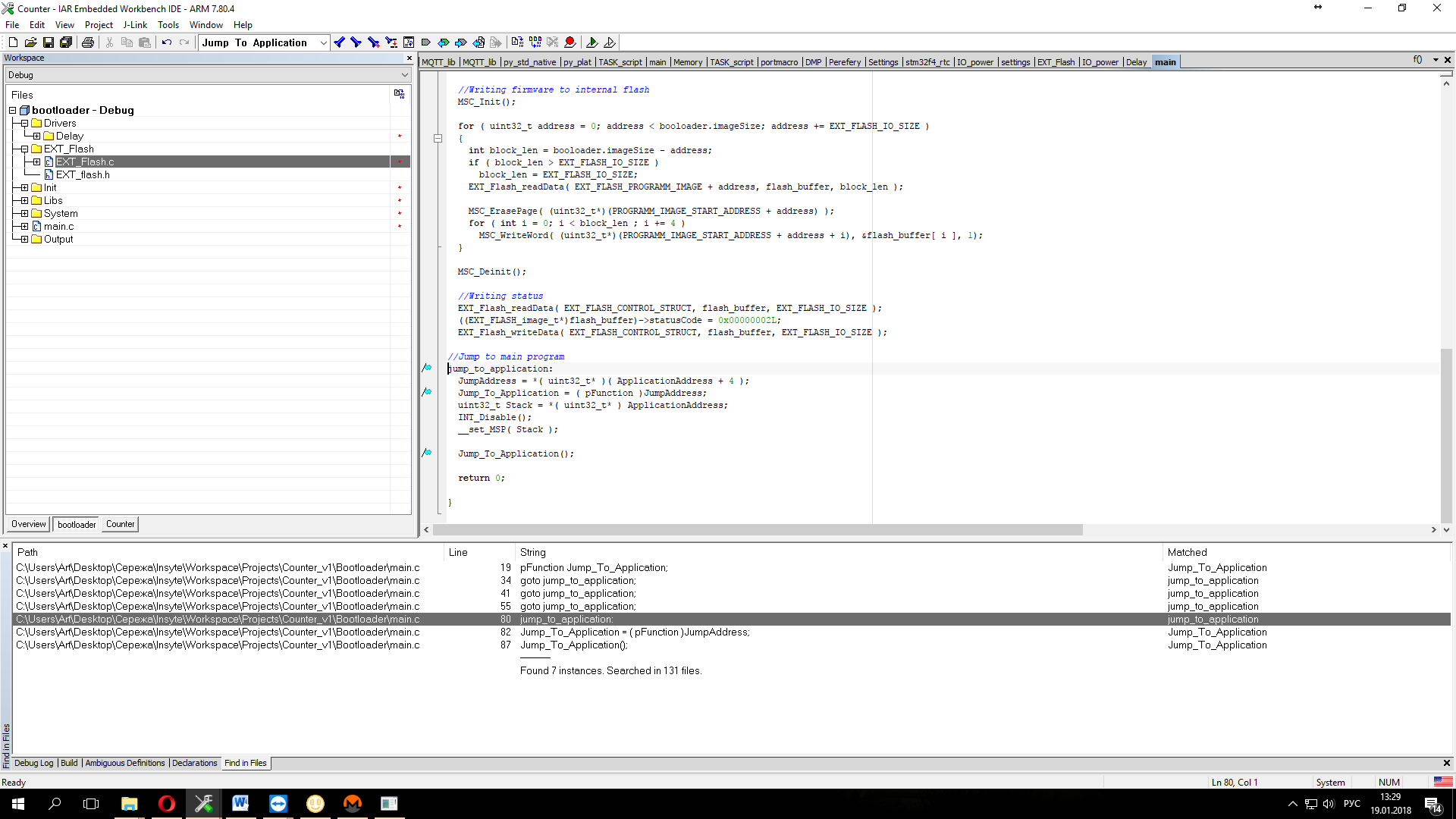
Рисунок 4.1 – настройки адресации памяти

Процесс загрузки прошивки начинается со скачивания прошивки основным ПО от различных каналов, таки как MQTT, SD карта памяти и USB. После закачивания прошивки в область внешней Flash памяти, производится расчет контрольной суммы прошивки и сравнение с текущим переданным значением. После чего в область памяти заголовка прошивки записывается magic code, его CRC контрольная сумма, размер и версия. После перезагрузки контроллера стартует загрузчик, который считывает magic code. Если magic code имеет значение COMMAND\_TO\_REPROGRAM, то производится повторная проверка CRC суммы и если все совпадает, что с помощью системных функций производится перепрограммирование памяти.



Листинг 4.1– работа загрузчика

Для запуска основного ПО используется лейбл jump\_to\_application, где создается функция по адресу начала ПО, которая в последствии и запускается.



Листинг 4.2 – переход в начало основного ПО

После перехода в основное ПО, необходимо переопределить адрес вектора прерывания.

5 Тестирование стабильности работы Ethernet соединения

Тестирование соединения необходимо для учета скорости, объема и периодичности отправки сообщений.

Тестирование будем производить на физическом и канальном уровне модели OSI. Для тестирования используется протокол IGMP и настраиваемая функция ping. Со стороны устроства реализован сервер echo, со стороны компьютера используется утилита Ping Tester Professional (рисунок).

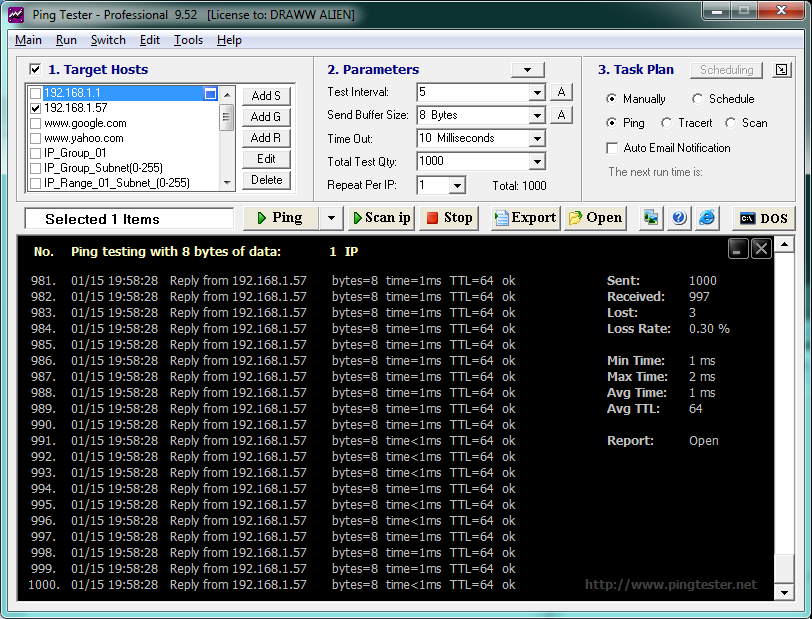


Рисунок 5.1 – утилита Ping Tester Professional

В качестве настраиваемых переменных используются:

- Интервал тестирования (Периодичность отправки сообщений) 1-10 мс

- Длина сообщения: 8-32000 байт

Выходным переменными являются:

- Количество потерянных пакетов в процентах от отправленных

- Среднее, минимальное и максимальное время пакета

Тестирование произведено по два раза для каждого периода и объема данных и взято их среднее значение.

Таким образом, получились следующие данные (Таблица 5.1, Рисунок 5.1).

Таблица 5.1– Данные тестирования стабильности работы Ethernet

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Период: T=1 мс | Длина сообщения, байт | | | | | | | | | | | | |
| 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32000 |
| Потери, % | 0.20 | 0.20 | 0.20 | 0.30 | 0.40 | 0.30 | 0.10 | 0.20 | 0.60 | 0.45 | 0.70 | 0.65 | 1.45 |
| Среднее время, мс | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 6 |
| Max время, мс | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 5 | 7 |
| Min время, мс | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 5 |
| Период: T=5 мс | Длина сообщения, байт | | | | | | | | | | | | |
| 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32000 |
| Потери, % | 0.10 | 0.10 | 0.10 | 0.10 | 0.25 | 0.30 | 0.30 | 0.35 | 0.25 | 0.25 | 0.60 | 0.50 | 1.20 |
| Среднее время, мс | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 6 |
| Max время, мс | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 5 | 9 |
| Min время, мс | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 5 |
| Период: T=10 мс | Длина сообщения, байт | | | | | | | | | | | | |
| 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32000 |
| Потери, % | 0.20 | 0.40 | 0.10 | 0.20 | 0.15 | 0.30 | 0.20 | 0.20 | 0.40 | 0.40 | 0.65 | 0.60 | 1.50 |
| Среднее время, мс | 1 | 1 | 1 | 1 | 1,5 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 6 |
| Max время, мс | 1 | 1 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 3 | 5 | 7 |
| Min время, мс | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1,5 | 3 | 5 |
| Период: T=20 мс | Длина сообщения, байт | | | | | | | | | | | | |
| 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32000 |
| Потери, % | 0 | 0.10 | 0.10 | 0.15 | 0.20 | 0.20 | 0.30 | 0.20 | 0.50 | 0.50 | 0.60 | 0.55 | 1.60 |
| Среднее время, мс | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1,5 | 2 | 3 | 6 |
| Max время, мс | 1 | 1 | 1 | 1,5 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 5 | 7 |
| Min время, мс | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 5 |

Рисунок 5.1 – графики стабильности связи

Таким образом, из графиков получаем, что потери значительно растут после увеличения объема от 2 Кбайт. При этом среднее время приема – передачи пакета не зависит от периода отправления и так же в районе 2 Кбайт увеличивает свое значение.

Отсюда следует, что наиболее оптимальным является отправка посылок не больше 2 Кбайт с периодичностью от 20мс.

ЗАКЛЮЧЕНИЕ

В работе расписана структура проекта, описаны и разработаны недостающие функциональные блоки, рассмотрены их функции и особенности. Подготовлена тестовая модель и испытан сетевой интерфейс Ethernet, составлены графики ошибок, определена оптимальные параметры отправки данных