1. Автоматизация процесса разработки и отладки программ для микроконтроллеров и для созданных на их основе устройств интернета вещей является важной и острой задачей для разработчиков. Это вызвано, в первую очередь, ограниченностью интерфейсов микроконтроллеров для взаимодействия с пользователем и отсутствием высокоуровневых средств отладки.

2. Особо остро эти проблемы встают перед платформой визуальной аналитики и научной визуализации SciVi, разработанной сотрудниками кафедры МОВС. Это обусловлено тем, что в данной платформе микроконтроллеры используются не просто для исполнения единожды загруженных в них программ, а для для интерпретации пользовательских алгоритмов, передающихся на микроконтроллеры в процессе работы. Причём передаются алгоритмы в виде онтологий, сжатых с помощью набора программных средств EON, который так же разработан сотрудниками кафедры. Решению проблем автоматизации программирования микроконтроллеров и посвящена моя работа.

3. Объект исследования: автоматизация периферийных вычислений. Предмет исследований: средства платформы SciVi для организации онтологически-управляемых периферийных вычислений.

4. Цель ВКР, в рамках котрой была проведена НИР – разработка требуемых платформой SciVi программных средств для автоматизации программирования микроконтроллеров.

5. А цель самой НИР: Разработать программный модуль с высокоуровневым интерфейсом, позволяющий удобно и без необходимости ручной настройки сохранять и считывать информацию из энергонезависимой памяти микроконтроллеров, в соответствии с требованиями платформы SciVi. Платформе SciVi такой модуль необходим для повышения уровня автономности и отказоустойчивости используемых устройств интернета вещей за счёт автоматизации восстановления состояния этих устройств после их перезагрузки. Отмечу, что в микроконтроллерах обычно используются электрически стираемые перепрограммируемые постоянные запоминающие устройства (Electrically Erasable Programmable Read-Only Memory, EEPROM), позволяющие стереть сохранённые данные, а затем записать новые.

6. Для достижения цели НИР были поставлены следующие задачи:

   (a) Составить требования к необходимому программному модулю.

   (b) Исследовать существующие средства для работы с энергонезависимой памятью и, в частности, EEPROM микроконтроллеров.

   (c) При возможности, выбрать одно из таких средств для использования в качестве основы разрабатываемого модуля.

   (d) Разработать программный модуль для работы с EEPROM микроконтроллеров, соответствующий всем поставленным требованиям.

(e) Провести тестирование и отладку разработанного программного модуля.

(f) Интегрировать разработанный модуль в платформу SciVi.

7. Перед необходимым платформе SciVi программным модулем были поставлены следующие требования:

   (a) Наличие возможности сохранять и считывать данные произвольной структуры из EEPROM. Это основное назначение модуля.

   (b) Для достижения необходимого уровня удобства в использовании, обращение к данным в EEPROM должно производиться по некоторым, заданным пользователем, идентификаторам, без необходимости ручных манипуляций с EEPROM адресами.

   (c) Минимизация количества операций записи в EEPROM. Общее количество циклов перезаписи для EEPROM ограничено и обычно составляет от 100,000 до 1,000,000 циклов, поэтому количество операций записи необходимо сводить к минимуму.

   (d) Программный модуль должен выполняться на микроконтроллерах серии ESP8266, так как именно их использует SciVi и, по возможности, на платформе Arduino, так как она пользуется наибольшей популярностью в мире.

8. Микроконтроллеры серии ESP8266 имеют важную особенность: в ESP8266 в качестве EEPROM используется flash-память, позволяющая стирать данные только большими блоками. В то время, как во многих других микроконтроллерах EEPROM позволяет стирать отдельные байты, не уменьшая ресурс остальных. Причём в ходе проведённой работы при анализе исходного кода библиотек для ESP8266 было установлено, что размер стираемого блока данных равен всему объёму памяти, доступному пользователю. Т.е. для обновления даже одного байта в ESP8266 необходимо, фактически, перезаписать в EEPROM все данные.

9. На первом этапе работы были проанализированы наиболее популярные, из существующих, решения для управления энергонезависимой памятью микроконтроллеров. Первое из них – стандартная библиотека, входящая в состав набора инструментов для программирования микроконтроллеров Arduino IDE. Она предоставляет низкоуровневые функции для чтения и записи в EEPROM и обёртку вокруг функции записи, производящую перезапись только в случае, если записываемы данные не совпадают с уже находящимися в EEPROM. Однако все эти функции требуют ручного указания адреса для записи или чтения, что, очевидно, не соответствует поставленным требованиям.

10. Второе рассмотренное решение - библиотека EEManager. Данная библиотека включает в себя механизм отложенной записи, не позволяющий производить перезапись данных в EEPROM слишком часто. И механизм ключа первой записи, позволяющий упростить процесс записи начального значения в

EEPROM и последующего чтения из него. Но при этом данная библиотека также требует явно указывать адреса EEPROM для взаимодействия, что так же нарушает поставленные требования.

11. Последняя из рассмотренных библиотек – EEPROMWearLevel. Данная библиотека позволяет обращаться к блокам данных в EEPROM по их индексам, а не адресам. Это также не позволяет использовать библиотеку в нескольких независимых программных модулях из-за того, что множество индексов едино для всего EEPROM и не может быть разделено между модулями, однако EEPROMWearLevel подходит к этой цели ближе всех других рассмотренных библиотек. Кроме того, библиотека содержит ряд механизмов, способных значительно уменьшить износ EEPROM с побайтовой перезаписью. Однако, при использовании с flash памятью эти механизмы наоборот только ускорят износ, кроме того, их реализация содержит платформозависимый код, выполнение которого на микроконтроллерах ESP8266 невозможно.

12. На основе проведённого анализа был сделан вывод о необходимости создания собственной библиотеки для работы с EEPROM. На первых этапах проектирования было решено в качестве идентификаторов блоков данных использовать текстовые имена. А для уменьшения вероятности коллизий этих имён, разделить все данные, хранящиеся в EEPROM на разделы – по сути, пространства имён для блоков данных.

13. С учётом этого была составлена следующая диаграммы прецедентов использования разрабатываемой библиотеки, на основе которой производилось дальнейшее проектирование библиотеки.

14. Затем была разработана структура хранения данных в EEPROM. В ней стоит отметить наличие системного раздела памяти, которых хранит информацию об остальных разделах и о EEPROM в целом. Остальные же разделы описываются в EEPROM так же, как любые пользовательские данные – за счёт их помещения в EEPROM-переменные.

15. Наконец, была составлена диаграммы классов разрабатываемом библиотеки, включающая в себя классы для сущностей EEPROM-переменной, раздела EEPROM и описания EEPROM в целом. Первая версия библиотеки была реализована в соответствии с разработанными моделями. Однако при её тестировании было явлено, что несмотря на то, что её логика работает в соответствии с ожиданиями, её интерфейс является не достаточно высокоуровневым для достижения необходимого уровня удобства разработки.

16. В качестве последнего, достигнутого на данный момент результата, были составлены общие требования по переработке и улучшению библиотеки.