Федеральное государственное бюджетное образовательное учреждение
высшего образования «Российский государственный
аграрный университет - МСХА имени К.А. Тимирязева»

На правах рукописи

СИДОРЕНКО МИХАИЛ СЕРГЕЕВИЧ

МЕТОД И ЭЛЕКТРОТЕХНИЧЕСКИЕ СРЕДСТВА МОНИТОРИНГА НАЧАЛА ОТЕЛА У КОРОВ

05.20.02 – электротехнологии и электрооборудование в сельском хозяйстве

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата технических наук

Научный руководитель:
доктор технических наук,
доцент Иванов Ю.Г.

Москва − 2016

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 5

ГЛАВА 1. АНАЛИЗ СОСТОЯНИЯ ПРОБЛЕМЫ. ЦЕЛИ И ЗАДАЧИ ИССЛЕДОВАНИЯ. 10

1.1. Проблема дистоции у коров. 10

1.2. Анализ методов и технических средств определения отела у коров. 14

1.3. Цели и задачи исследований. 22

ГЛАВА 2. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ ПАРАМЕТРОВ ЭТС ДЛЯ МОНИТОРИНГА НАЧАЛА ОТЕЛА У КОРОВ 23

2.1. Классификация явлений, происходящих с хвостом животного. 23

2.2. Предвестники родов и требования к первичным электротехническим средствам для их идентификации 24

2.3. Физические параметры при родовом акте, пригодные для идентификации электротехническими средствами 26

2.4. ЭТИП регистрации мышечного напряжения корня хвоста животного. 27

2.5. Математическая модель ЭТИП мышечного напряжения корня хвоста на основе измерения давления на чувствительный элемент. 28

2.6. Математическая модель ЭТИП мышечного напряжения корня хвоста на основе измерения деформации чувствительного элемента. 34

2.7. Математическая модель ЭТИП мышечного напряжения корня хвоста животного на основе измерения относительного удлинения длины окружности тензочувствительного элемента, наполненного электропроводящим раствором. 37

2.8. Алгоритм обработки сигналов ЭТИП мышечного напряжения для идентификации начала родов у коров 39

ГЛАВА 3. ПРОГРАММА, МЕТОДИКА И ЭКСПЕРИМЕНТАЛЬНАЯ УСТАНОВКА ДЛЯ ПРОВЕДЕНИЯ ИССЛЕДОВАНИЙ. 44

3.1. Методика исследований ЭТС на модели хвоста животного. 45

3.2. Методика исследований ЭТС при измерении мышечного напряжения на руке человека. 46

3.3. Методика исследований ЭТС при измерении мышечного напряжения на хвосте животного. 47

3.4. Методика исследований ЭТС при родах животного. 48

ГЛАВА 4. ЭЛЕКТРОТЕХНИЧЕСКИЕ СРЕДСТВА ДЛЯ МОНИТОРИНГА НАЧАЛА ОТЕЛА У КОРОВ. 50

4.1. Аппаратная часть. 53

4.1.1. Устройство ЭТИП. 53

4.1.1.1. Схема электрическая структурная. 53

4.1.1.2. Обоснование выбора элементной базы. 55

4.1.1.3. Схема электрическая принципиальная. 56
4.1.1.4. Расчет надежности. 58
4.1.1.5. Расчет продолжительности работы от батареи. 62
4.1.2. Устройство сопряжения с ПЭВМ. 64
4.1.2.1. Схема электрическая структурная. 64
4.1.2.2. Обоснование выбора элементной базы. 65
4.1.2.3. Схема электрическая принципиальная. 67
4.1.2.4. Расчет надежности. 68
4.2. Программная часть. 71
4.2.1. Устройства ЭТИП. 71
4.2.1.1. Обоснование выбора средств и методов программирования управляющего микроконтроллера. 71
4.2.1.2. Описание логической структуры микропрограммы управляющего микроконтроллера. 72
4.2.1.3. Описание функций программных модулей управляющего микроконтроллера. 74
4.2.1.4. Протокол обмена информацией с устройством сопряжения с ПЭВМ. 78
4.2.2. Устройство сопряжения с ПЭВМ. 80
4.2.2.1. Обоснование выбора средств и методов программирования управляющего микроконтроллера. 80
4.2.3. Описание логической структуры микропрограммы управляющего микроконтроллера. 81
4.2.3.1. Описание функций программных модулей управляющего микроконтроллера. 82
4.2.3.2. Протокол обмена информацией с ПЭВМ. 86
4.2.4. Программа интерпретации данных для ПЭВМ. 88
4.2.4.1. Обоснование выбора средств и методов программирования. 88
4.2.4.2. Описание логической структуры программы. 88
4.2.4.3. Описание функций программных модулей. 91
ГЛАВА 5. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ. 95
5.1. Результаты исследований ЭТС на модели хвоста животного. 95
5.2. Результаты исследований ЭТС при измерении мышечного напряжения на руке человека. 95
5.3. Результаты исследований ЭТС при измерении мышечного напряжения на хвосте животного. 98
5.4. Результаты исследований ЭТС при родах животного. 100
ГЛАВА 6. ДОСТОВЕРНОСТЬ ДЕТЕКТИРОВАНИЯ РОДОВЫХ СХВАТОК И АППАРАТНАЯ НАДЕЖНОСТЬ ЭЛЕКТРОТЕХНИЧЕСКИХ СРЕДСТВ МОНИТОРИНГА НАЧАЛА ОТЕЛА У КОРОВ. 102

ГЛАВА 7. ЭКОНОМИЧЕСКАЯ ЭФФЕКТИВНОСТЬ ПРИМЕНЕНИЯ ЭТС ДЛЯ МОНИТОРИНГА НАЧАЛА ОТЕЛА У КОРОВ. 106
7.1. Факторы экономической эффективности применения ЭТС для определения отела у коров. 106
7.2. Расчет стоимости ЭТС и определение затрат на его обслуживание. 110
7.3. Расчет экономической эффективности применения ЭТС. 114
ОСНОВНЫЕ ВЫВОДЫ 118
СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ 120
СПИСОК ЛИТЕРАТУРЫ 121
ПРИЛОЖЕНИЯ 136
Приложение А: Документы об использовании результатов исследований. 136
Приложение Б: Патенты и свидетельства. 140
Приложение В: Структурные и принципиальные схемы. 142
Приложение Г: Листинги. 148
ЛИСТИНГ МИКРОПРОГРАММЫ ЭТИП 148
ЛИСТИНГ МИКРОПРОГРАММЫ УСТРОЙСТВА СОПРЯЖЕНИЯ С ПЭВМ 200
ЛИСТИНГ СПО 239

ВВЕДЕНИЕ

Актуальность темы исследования. Важным резервом повышения эффективности производства молока является оперативное реагирование на возникающие производственные проблемы путем применения средств и систем контроля и управления технологическими процессами.

С увеличением размеров молочных ферм и возрастанием рабочей нагрузки на обслуживающий персонал возникает необходимость автоматизации дистанционного контроля за каждым животным в режиме реального времени. Известно, что коровы посредством поведения, расположения тела и других физиологических проявлений "подают сигналы", по которым можно судить об их здоровье и физиологическом состоянии. Своевременная регистрация этих сигналов и оказание помощи животным позволит повысить рентабельность фермы.

Одной из важнейших причин недополучения молодняка и снижения молочной продуктивности является осложнения течения родового акта у коров из-за недостоточно точного выявления момента наступления родов. При этом зачастую возникают патологические роды, или дистоция, ущерб от которой достигает существенных размеров. По оценкам специалистов в США, ущерб от смертности КРС при родах составляет около 1 млрд. долларов ежегодно. Изучение этой проблемы показывает, что на практике отсутствуют средства оперативного контроля и оповещения обслуживающего персонала, которые бы позволили оказать своевременное родовспоможение, в связи с чем разработка электро-технических средств (ЭТС), обеспечивающих мониторинг соответствующих признаков у коров с дистанционной передачей сообщения обслуживающему персоналу о начале родов в режиме реального времени является актуальной задачей.

Степень разработанности темы. Несмотря на насущную потребность предприятий крупного рогатого скота в средствах определения отела у коров, разработке технических средств для этих целей посвящено немного публикаций, например, работы О.В. Ужик и Я.В. Ужик (Россия), Е. Малтц (США) и других.

Известны используемые за рубежом технические средства, предназначенных для решения этой задачи, например, системы фирм «Medria Technologies» и «Deutsche Telekom» (Франция, Германия), «iVET» (Германия), «Radco» (Бельгия), «Databel Trading» (Бельгия), «MaGiiX» (США). Методы детектирования, используемые в этих технических решениях, не позволяют однозначно определить явления начала родов, при этом отечественные разработки отсутствуют.

Целью работы является разработка метода и электротехнических средств мониторинга начала отела у коров, обеспечивающих оповещение персонала фермы о начале родовых схваток в режиме реального времени для своевременного родовспоможения.

Для достижения поставленной цели определены следующие задачи:

1. Проанализировать научные предпосылки и тенденции развития методов и технических средств для определения начала родов у коров.
2. Определить признаки родового акта коровы, которые можно представить в виде сигналов, пригодных для регистрации электротехническими средствами дистанционной передачи сообщений персоналу фермы в режиме реального времени.
3. Разработать математические модели вариантов электротехнических измерительных преобразователей напряжения мышц корня хвоста животного.
4. Разработать, изготовить и исследовать варианты электротехнических измерительных преобразователей напряжения мышц корня хвоста животного и выбрать наиболее предпочтительный вариант.
5. Разработать алгоритм обработки сигналов электротехнических измерительных преобразователей напряжения мышц корня с помощью специализированного программного обеспечения для достоверной идентификации начала родов у коров.
6. Разработать программу, методики и установки для проведения экспериментальных исследований.

7. Создать комплект электротехнических и программных средств для дистанционного мониторинга начала отела у коров и передачи сообщений персоналу в режиме реального времени.
8. Провести экспериментальные исследования электротехнических средств.
9. Произвести производственную проверку электротехнических средств.
10. Провести оценку экономической эффективности применения электротехнических средств.

Научная новизна. Выполненные исследования позволили получить совокупность новых положений и результатов, заключающихся:

- в разработке миографического метода для контроля признаков начала родового акта коровы и их представления в виде сигналов, пригодных для регистрации электротехническими средствами;

- в разработке математических моделей вариантов электротехнических измерительных преобразователей напряжения мышц корня хвоста животного и алгоритма обработки сигнала для идентификации родового акта коровы;

- в разработке, изготовлении и апробации электротехнического и программного комплекса, обеспечивающего высокую достоверность обнаружения родового акта, с дистанционным мониторингом в режиме реального времени.

Теоретическая и практическая значимость работы. Разработаны новые электротехнические средства, обеспечивающие определение отела у коров в режиме реального времени и передачи сообщения обслуживающему персоналу с помощью современных телекоммуникационных средств связи. Производственная проверка ЭТС проведена на молочной ферме зоостанции РГАУ-МСХА имени К.А. Тимирязева. Техническая новизна разработанных ЭТС подтверждена патентом РФ (№134782, Система определения начала отела) и свидетельством о государственной регистрации программы для ЭВМ (№ 2015616713, Драйвер микросхемы радиоинтерфейса nRF24L01 для микроконтроллеров с ядром AVR.).

Методология и методы исследования. В ходе теоретических исследований были использованы методы математического анализа, математического и компьютерного моделирования, программирования. Использовались средства

микропроцессорной и компьютерной техники. Результаты экспериментов обрабатывались в соответствии с общепринятыми методиками с использованием программных продуктов MathCAD 14, Microsoft Office Excel 2007, Microsoft Visio 2007, AVR Studio 6, Visual Studio 2012.

Положения, выносимые на защиту:

- миографический метод мониторинга начала отела у коров, основанный на регистрации напряжения мышц корня хвоста, сопровождающего схватки и потуги животного, электротехническими средствами;

- математические модели измерительных преобразователей напряжения мышц корня хвоста животного в электрический сигнал;

- алгоритм обработки сигналов от электротехнического измерительного преобразователя напряжения мышц корня хвоста животного, обеспечивающего достоверную идентификацию начала родов у коров;

- конструкция и принцип работы измерительного преобразователя напряжения мышц корня хвоста животного на основе изменения электропроводности чувствительного элемента.

- комплект электротехнических и программных средства для мониторинга начала отела у коров.

Степень достоверности и апробация результатов подтверждена высокой сходимостью результатов теоретических и экспериментальных исследований, положительными результатами опытов и производственных испытаний, разработанных технических и программных решений. По материалам диссертации опубликованы семь статей, из них пять - в журналах, рекомендованных ВАК Министерства образования и науки РФ. Получен патент РФ и свидетельство о государственной регистрации программы для ЭВМ.

Основные положения диссертационной работы были доложены, обсуждены и одобрены на: Всероссийской конференции студентов, аспирантов и ученых с международным участием «Основные направления развития техники и технологии в АПК, легкой и пищевой промышленности» (Княгинино, НГИЭИ, 2012 г.);

Международной научно–практической конференции, посвященной 85–летию академика К.У. Медеубекова «Зоотехническая наука Казахстана: прошлое, настоящее, будущее», г. Алматы в 2014 г.

17-й Международной научно-практической конференции «Инновационные технологии и технические средства производства продукции животноводства с интеллектуальными системами управления механизированными процессами» (Москва, ВНИИМЖ, 2014 г.);

9-ой Международной научно-технической конференции "Энергообеспечение и энергосбережение в сельском хозяйстве" (Москва, ВИЭСХ, 2014 г.);

18-й Международной научно-практической конференции «Стратегия развития механизации и автоматизации животноводства на период до 2030 года» (Москва, ВНИИМЖ, 2015 г.).

ГЛАВА 1. АНАЛИЗ СОСТОЯНИЯ ПРОБЛЕМЫ. ЦЕЛИ И ЗАДАЧИ ИССЛЕДОВАНИЯ.

1.1. Проблема дистоции у коров.

Значительный вклад в интенсификацию воспроизводства крупного рогатого скота внесли ученые В.С. Авдеенко [67, 75], Г.П. Дюльгер [51, 91, 104, 110, 112, 114, 115], Д.В. Кривенко [61, 62, 67], Д.Д. Логвинов [66], А.Г. Нежданов [51, 81, 82, 83, 84], В.Я. Никитин [17, 86, 100], В.А. Павлов [90], А.М. Семиволос [106], А.П. Студенцов [16, 17], В.В. Храмцов [51], В.С. Шипилов [16, 17] и другие. Имеющиеся научные достижения по разным причинам не в должной мере используются на практике. В патологии крупного рогатого скота одним из узких мест остаются осложненные роды.

Семиволос А.М. отмечает, что акушерско-гинекологические болезни у коров регистрируются на протяжении всего года [106]. Кривенко Д.В. отмечает, что патология родов сопровождается ростом числа мертворожденных телят: от 4,8% - при слабых нарушениях до 50,7 % - при хирургическом вмешательстве, а также обусловливающие многочисленные осложнения у новорожденного и матери. При этом частота осложненных родов у коров составляет около 7%, что говорит о распространенности проблемы. Наиболее часто этим осложнением подвержены высокопродуктивные коровы (28,1%), первотелки (8,61%) и коровы в возрасте более пяти отелов, около 6%. Основными причинами осложненных родов являются: нарушение течения беременности (38%), аборты (53%), фетоплацентарная недостаточность (75%). При осложнении родов происходит нарушение течения инволюционных процессов, возрастает частота послеродовых осложнений (54%), снижается оплодотворяемость от первого осеменения (27%) [62].

Одной из наиболее острых проблем на фермах являются патологические роды, или дистоция - это роды, течение которых нарушено по разным причинам, и проведение которых требует оперативных вмешательств. По оценкам специалистов, общая годовая стоимость потерь хозяйств (для США) вследствие

бесплодия, выкидышей, мертворождения, дистоции и других проблем в репродуктивной системе коров составляет от 441 до 502 млн. долларов для мясного, и от 473 до 484 млн. долларов для молочного скотоводства. Эти потери, в общей совокупности, составляют ежегодно около 1 млрд. долларов. Они в шесть раз превышают потери от заболеваний дыхательной системы. Три четверти этих потерь возникают вследствие бесплодия, дистоции и слабых телят, которые не выживут в первые 24 часа жизни [150].

Известно, что отел должен проходить под наблюдением, так как нельзя заранее предугадать, как будут протекать роды. В начале родов важным моментом является определение положения, позиции и членорасположение плода. При нормальных родах работник должен следить за состоянием промежности и, в случае растяжения, придерживать ее рукой, чтобы предотвратить разрывы. Если после выхода головки и ножек плод задерживается в родовых путях, необходимо подтягивать его за ножки, захватив их руками или веревочными петлями (рисунок 1) [16].

Рисунок 1 – Помощь роженице

При патологии родов акушерскую помощь оказывают чаще всего непосредственно во время родов. Обычно она бывает неотложной, всякое промедление может стоит жизни плоду и матери.

Оказание своевременной и квалифицированной помощи при родах у коровы является основной составляющей успешных родов. Одной из основных причин высокого процента неблагополучных родов у молочных коров, по данным В.А. Павлова, является недостаточно точное выявление момента наступления родов.

На практике, как правило, это обеспечивается простым наблюдением персонала фермы, что требует значительных трудозатрат. Н.Ю. Терентьева отмечает, что у высокоудойных коров проявляющиеся предвестники родов имеют большой временной разброс – от 1 до 9 суток перед отелом, и установить момент родов с точностью до суток достаточно тяжело, в связи с чем своевременно перевести роженицу в индивидуальный станок не всегда удается, что способствует повышению процента патологических родов. То же и в оказании акушерской помощи роженицам – при затрудненном прогнозировании начала родов акушерская помощь чаще оказывалась с запозданием, что приводит к различным осложнениям течения послеродового периода. У среднепродуктивных коров большая часть предвестников родов наблюдается за 1-2 суток перед отелом, тем самым несколько облегчается выявление животных в предродовой периоде [122].

Особенностями производства молока на современных молочных фермах является необходимость минимизации трудозатрат и оперативное реагирование на возникающие производственные проблемы. С увеличением размеров молочных ферм и возрастанием рабочей нагрузки на фермера возникает необходимость автоматизации дистанционного контроля за каждым животным в режиме реального времени. Своевременное выявление событий, требующих вмешательства фермера, позволит повысить рентабельность фермы.

Известно, что коровы посредством поведения, расположения тела и других физиологических проявлений "подают сигналы", по которым можно судить об их здоровье и физиологическом состоянии. Установление признака начала родов у

коровы для мониторинга техническими средствами и дистанционной передаче этой информации в режиме реального времени обслуживающему персоналу является актуальной задачей.

Проведение исследований в выбранном направлении исследований мы начали с изучения того, что сделано известными учеными в этой и близких областях науки.

Значительный вклад в развитие электротехнологий, механизации и автоматизации технологических процессов для животноводства внесли И.Ф. Бородин [3, 4], С.В. Вендин [11, 12, 13, 14, 15], И.И. Гришин [42, 43, 44, 45], Л.П. Карташов [74], В.В. Кирсанов [55], И.Н. Краснов [59], В.Р. Краусп [60], Н.Н. Курзин [63, 64, 65], С.В. Мельников [71, 94], Н.М. Морозов [76], Р.М. Славин [118], Д.С. Стребков [120], Ю.А. Судник [4], В.И. Сыроватка [2], В.Ф. Ужик [125, 126, 129], А.И. Учеваткин [134], Ю.А. Цой [2, 139], D. Armstong [143] и другие.

Широкие исследования по созданию электромеханизированных технологий в животноводстве проведены в ВИЭСХ, ВНИИМЖ, ВНИИКОМЖ, ВИЖ, С-ЗНИИМЭСХ, Белгородской ГСХА, Вятской ГСХА, МГАУ им. В.П. Горячкина, Оренбургском ГАУ, РГАУ-МСХА им. К.А. Тимирязева, Санкт - Петербургском ГАУ, ПО «Кургансельмаш» и др.

Значительный вклад в области разработки электротехнических средств и электротехнологий для молочного животноводства внесли зарубежные фирмы «DeLaval» (Швеция), «ГЕА Фарм Технолоджиз» (Германия), «Fullwood» (Великобритания), «SAC» (Дания), «Milklein» (Италия) и др.

Изучение работ предшественников показывает, что готовые решения для достижения поставленной цели отсутствуют.

1.2. Анализ методов и технических средств определения отела у коров.

Система Vel'Phone фирм Medria Technologies, Deutsche Telekom (Франция, Германия), рисунок 2. Принцип работы основан на мониторинге температуры тела во влагалище животного. Во время родового акта плод, движущийся по родовым путям, выталкивает датчик из влагалища и показания температуры при этом резко изменяются. Это событие обнаруживается системой. Недостатком системы является: 1 - относительная сложность установки датчика, требующая определенной квалификации персонала; 2 - система не отслеживает возможные трудные роды (дистоцию); 3 - при установке датчика существует вероятность занесения инфекции, вызывающей заболевания животного (эндометрит). Это может привести к потере коровы и/или теленка, что влечет за собой серьезный материальный ущерб.

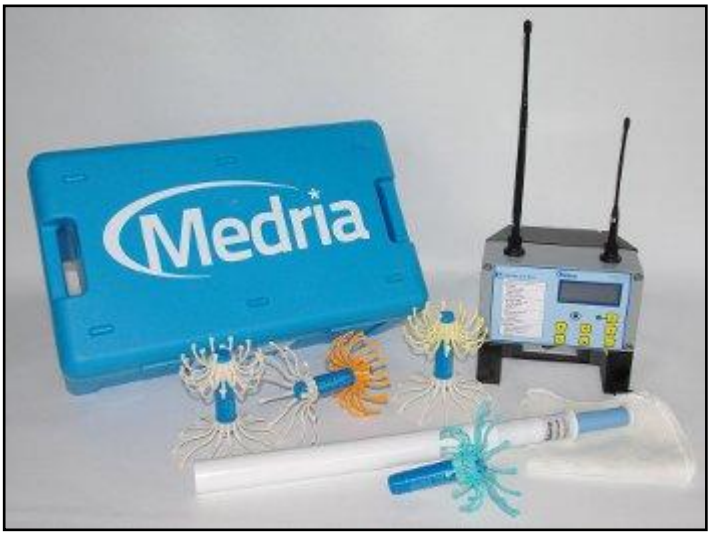


Рисунок 2 – Система Vel'Phone

Система iVET фирмы iVET (Германия). Принцип работы такой же, как и в предыдущей системе. Он основан на мониторинге температуры тела во влагалище животного. Во время родового акта плод, движущийся по родовым путям, выталкивает датчик из влагалища и показания температуры резко изменяются. Это событие обнаруживается системой. Недостатки системы те же, что и в предыдущей системе: 1 - относительная сложность установки датчика, требующая определенной квалификации персонала; 2 - система не отслеживает возможные

трудные роды (дистоцию); 3 - при установке датчика существует вероятность занесения инфекции, вызывающей заболевания животного (эндометрит). Это может привести к потере коровы и/или теленка, что влечет за собой серьезный материальный ущерб.

Система Radco фирмы Radco (Бельгия), рисунок 3. Принцип работы, как и в системе Vel'Phone фирм Medria Technologies, Deutsche Telekom, основан на мониторинге температуры тела во влагалище животного и изменения ее во время родового акта. Это событие обнаруживается системой. Недостатки системы те же, что и в системе Vel'Phone: 1 - относительная сложность установки датчика, требующая определенной квалификации персонала; 2 - система не отслеживает возможные трудные роды (дистоцию); 3 - при установке датчика существует вероятность занесения инфекции, вызывающей заболевания животного (эндометрит). Это может привести к потере коровы и/или теленка, что влечет за собой серьезный материальный ущерб.

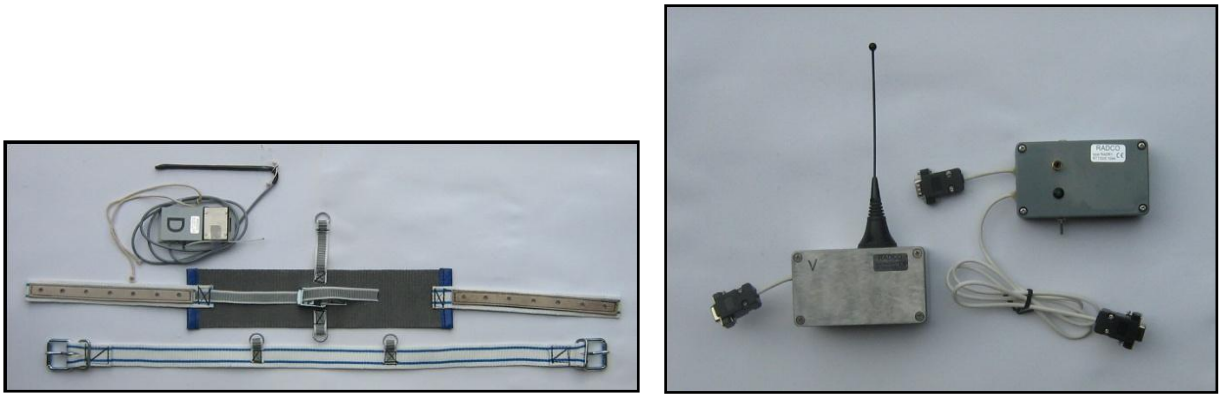


Рисунок 3 – Система Radco. Слева - зонд и система крепления. Справа - блок приема информации

Система Agrimonitor фирмы Databel Trading (Бельгия), рисунок 4. Принцип работы системы: мониторинг брюшных и маточных сокращений. Датчики системы располагаются на теле животного, отслеживая мышечные сокращения при потугах и схватках. Недостатком системы является сложность крепления датчиков к животному и сложность обеспечения их надежной фиксации во время работы в строго заданных местах на теле.

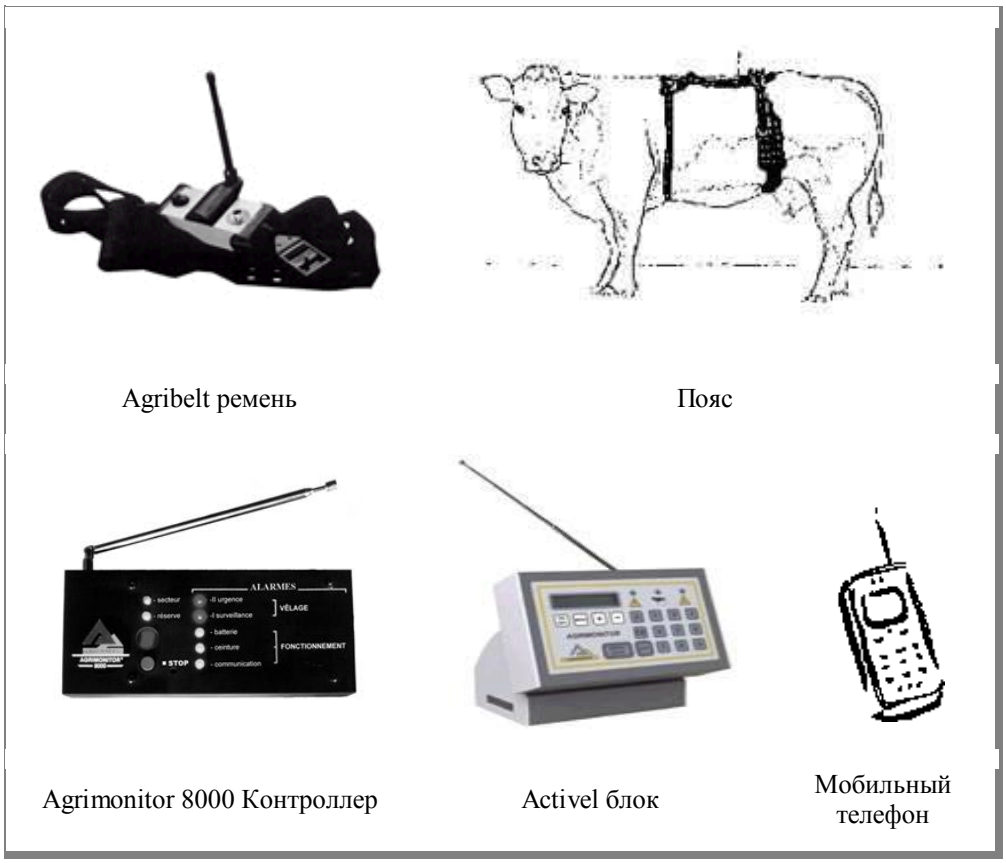


Рисунок 4 – Система Agrimonitor

Система MaGiiX фирмы MaGiiX (США), рисунок 5. Принцип работы системы: мониторинг измерения температуры тела животного. Датчик вводится в ретикулярную ткань животного. Существенным недостатком системы является необходимость хирургического вмешательства для введения датчика. Работа системы требует высококвалифицированного персонала и времени на восстановление животного после операции. Извлечение датчика так же представляет серьезную проблему.

Рисунок 5 – Система MaGiiX

Экспериментальная система (рисунок 6), основанная на принципе мониторинга количества шагов животного и времени проведенного им лежа (США). Пат. US 2010/0331739 A1 США. Method and system for predicting calving / E. Maltz. Недостатком метода является оценка состояния родового акта по косвенным признакам, что влечет за собою большую вероятность ошибки.

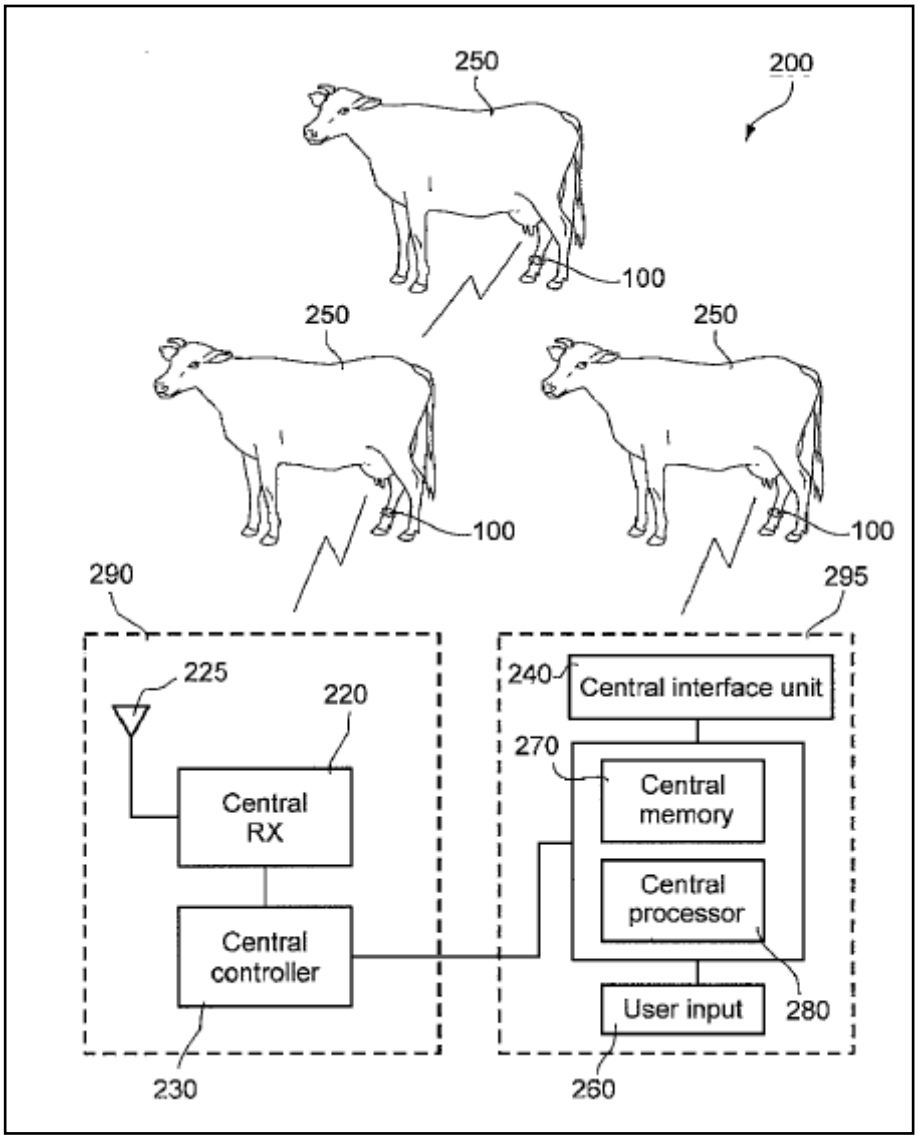


Рисунок 6 – Экспериментальная система определения отела, основанная на принципе мониторинга количества шагов животного и времени проведенного им лежа. Рисунок взят из патента US 2010/0331739 A1 США. Method and system for predicting calving / E. Maltz

Способ регистрации начала и завершения отела коровы (Рисунок 7) основан на принципе мониторинга изменения размеров крупа животного в результате движения плода в процессе родов (Россия). Пат. RU 2011/2454202 C1

Россия. Способ регистрации начала и завершения отела коровы / О.В. Ужик, Я.В. Ужик. Недостатком системы является сложность крепления датчиков к животному и обеспечение их надежной фиксации во время работы в строго заданных местах на теле.

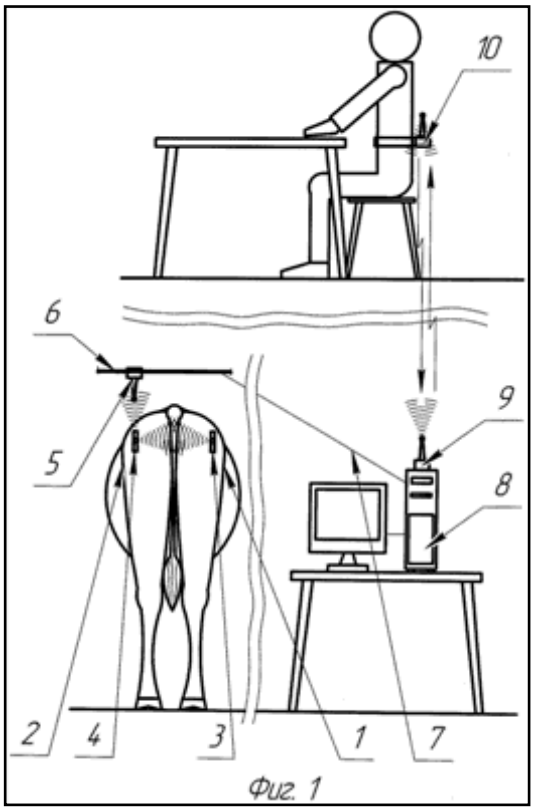


Рисунок 7 – Способ регистрации начала и завершения отела коровы. Рисунок взят из патента RU 2011/2454202 C1 Россия. Способ регистрации начала и завершения отела коровы / О.В. Ужик, Я.В. Ужик

Система видеонаблюдения в коровнике (Россия). Принцип работы системы заключается в мониторинге оператором за состоянием животных посредствам видеокамер. Недостаток системы заключается в необходимости постоянного мониторинга оператором за животными, что не слишком сильно отличается от непосредственного визуального наблюдения за животными в боксе. Проявления затруднений при родовом акте не всегда можно определить визуально.

Система Alert'Vel фирмы ALB Innovation (Франция), рисунок 8. Принцип работы системы: мониторинг положения хвоста животного. Во время родового

акта хвост животного находится в "приподнятом" состоянии гораздо дольше, чем при уринации и дефекации. Недостатком данной системы является оценка только "приподнятого" положения хвоста, однако корова может рожать и в положении лежа. В этом случае хвост не приподнят, а повернут в сторону. Такое состояние не всегда детектируется данной системой. Так же существует ряд заболеваний (эндометрит), при которых хвост практически всегда находится в приподнятом состоянии, что тоже может быть неадекватно воспринято системой.

Рисунок 8 – Система Alert'Vel

Анализ накопленного материала позволил составить сравнительные характеристики различных вариантов ЭТС для определения родового акта коровы (таблица 1).

Таблица 1 - Сравнительные характеристики вариантов ЭТС

№ п/п	Методы контроля	Достоинства	Недостатки	Существующий аналог или патент
1	Тепловизионное распознавание образа родового акта	Использование стандартного оборудования; прямое определение родового акта	Сложность реализации компьютерного распознавания образа родового акта, дорогостоящее оборудование, сложность эксплуатации, низкая надежность оборудования	Аналог или патент не найдены
2	Мониторинг изменения температуры животного и температуры внешней среды	Прямое определение родового акта	Возможны ошибки определения родового акта при патологических родах	Система Vel'Phone
3	Определение ускорения выпадения датчика из влагалища	Прямое определение родового акта	Возможны ошибки определения родового акта при патологических родах	Система iVET
4	Мониторинг количества шагов животного и времени, проведенного им лежа	Простота реализации системы	Косвенное определение родового акта; работа в составе с другими системами определения предвестников родов	Пат. US 2010/0331739 A1
5	Определение центра стада, положения животного относительно этого центра и времени, проводимого им в удалении от центра	Неограниченная зона покрытия системы	Сложность системы; корова должна находиться в стаде; работа в составе с другими системами определения предвестников родов	Система Agrimonitor
6	Определение сокращений мышц брюшного пресса (потуги)	Прямое определение родового акта	Работа в составе с другими системами определения предвестников родов	Система Agrimonitor
7	Определение сокращений мышц матки (схватки)	Прямое определение родового акта	Работа в составе с другими системами определения предвестников родов	Система Agrimonitor
8	Мониторинг положения корня хвоста животного в пространстве и времени	Прямое определение родового акта	Возможны ошибки из-за несоответствия относительного положения хвоста при родах стоя и лежа	Система Alert'Vel
9	Определение мышечного напряжения хвоста животного во время родового акта	Прямое определение родового акта	Необходимо рациональное техническое и программное решение ЭТС	Разрабатываемое ЭТС

Проведенные нами исследования на молочной ферме РГАУ-МСХА имени К.А. Тимирязева и фермы ЗАО «Агрофирма «Бунятино» Дмитровского района Московской области позволили установить, что в начале родов схватки (сокращении мышц матки) и потуги (сокращении мышц брюшной полости)

сопровождаются также напряжением мышц корня хвоста, который приподнимается.

Нами предложен новый метод, основанный на мониторинге сигналов мышечного напряжения корня хвоста животного, наблюдаемого во время начала родов у коров, для дистанционного оперативного информирования обслуживающего персонала в режиме реального времени.

Для реализации метода целесообразна разработка электротехнических средств, основанных на регистрации и анализе сигналов мышечного напряжения корня хвоста животного.

Из приведенной сравнительной характеристики видно, что системы определения родового акта сложны в своей реализации. На наш взгляд наибольший интерес представляют ЭТС, действие которых основано на явлениях, происходящих с хвостом животного. Для этого есть как минимум три причины: 1 − это системы прямого определения родового акта; 2 − достоверность их показаний сохраняется и при явлениях дистоции; 3 − они не причиняют стресс животному, так как устанавливаются снаружи на хвосте, а не внутри − во влагалище. Как известно, во время родового акта:

1 − хвост располагается по дуге от корня к кончику; 2 − хвост (возле корня) продолжает линию позвоночника независимо от положения животного (стоя или лежа); 3 − патологическое течение родов не влияет на мышечное напряжение корня хвоста [113].

Анализ показал, что одним из наиболее перспективных путей при разработке ЭТС для определения начала родов у коров является применение метода, основанного на регистрации мышечного напряжения коровы при родах. При этом начало родов сопровождается напряжением мышц корня хвоста, который отводится в сторону, освобождая путь для рождающегося теленка.

Анализ ЭТС для определения начала родов у коров показал перспективность ЭТС, основанных на регистрации изменения мышечных напряжений корня хвоста животного.

1.3. Цели и задачи исследований.

Для достижения поставленной цели определены следующие задачи:

1. Проанализировать научные предпосылки и тенденции развития методов и технических средств для определения начала родов у коров.
2. Определить признаки родового акта коровы, которые можно представить в виде сигналов, пригодных для регистрации электротехническими средствами дистанционной передачи сообщений персоналу фермы в режиме реального времени.
3. Разработать математические модели вариантов электротехнических измерительных преобразователей напряжения мышц корня хвоста животного.
4. Разработать, изготовить и исследовать варианты электротехнических измерительных преобразователей напряжения мышц корня хвоста животного и выбрать наиболее предпочтительный вариант.
5. Разработать алгоритм обработки сигналов электротехнических измерительных преобразователей напряжения мышц корня с помощью специализированного программного обеспечения для достоверной идентификации начала родов у коров.
6. Разработать программу, методики и установки для проведения экспериментальных исследований.
7. Создать комплект электротехнических и программных средств для дистанционного мониторинга начала отела у коров и передачи сообщений персоналу в режиме реального времени.
8. Провести экспериментальные исследования электротехнических средств.
9. Произвести производственную проверку электротехнических средств.
10. Провести оценку экономической эффективности применения электротехнических средств.

ГЛАВА 2. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ ПАРАМЕТРОВ ЭТС ДЛЯ МОНИТОРИНГА НАЧАЛА ОТЕЛА У КОРОВ

2.1.Классификация явлений, происходящих с хвостом животного.

Физиологические проявления у коров, сопровождаемые явлениями, происходящими с хвостом с точки зрения их детектирования, могут быть классифицированы так, как это представлено на рисунке 9.

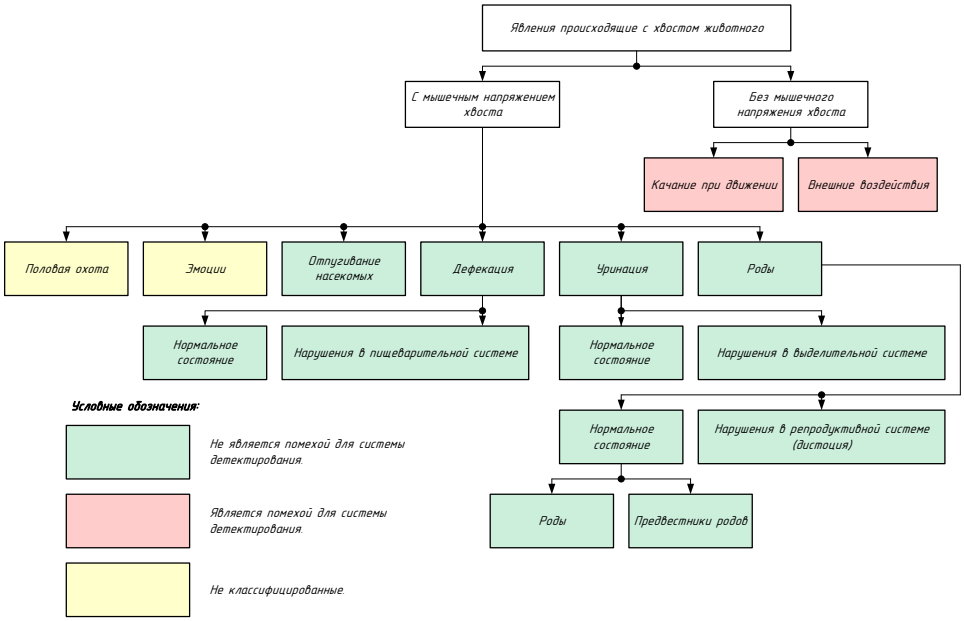


Рисунок 9 – Классификация явлений, происходящих с хвостом коровы

Из рисунка видно, что некоторые явления являются помехой для ЭТС. Другие не подверглись классификации в связи со сложностью их интерпретации - это ЭТС «Эмоции» и «Половая охота». Явлениям, не подвергшимся классификации, можно найти потенциальную возможность практического применения в ветеринарии и животноводстве при условии проведения отдельного комплексного исследования. Например, определение половой охоты коров при использовании быка в стаде не рассматривается в настоящей работе. Стоит заметить, что ЭТС для определения отела у коров имеет техническую возможность для их детектирования.

Поведение хвоста животного зависит от физиологического состояния животного. Например, при уринации или дефекации животное поднимает хвост и

удерживает его в поднятом состоянии некоторое время. При этом поведение хвоста отражает состояние мочеполовой системы (например, эндометрит - рисунок 10). Также поведение хвоста кардинально меняется в процессе родов (рисунок 10). За несколько часов до родового акта он приобретает положение, показанное на рисунке 10. Мышцы корня хвоста начинают сильно напрягаться. Интервалы между напряжением мышц сокращаются и плавно переходят в потуги.

Анализ показывает, что функционирование ЭТС может быть основано на оперативном анализе мышечной активности корня хвоста животного.

Рисунок 10 - Слева: положение хвоста больного эндометритом (выделено красным) и здорового (выделено зеленым) животного. Справа: положение хвоста животного (выделено зеленым) за четыре часа до родового акта

2.2. Предвестники родов и требования к первичным электротехническим средствам для их идентификации

Для достоверного определения предвестников родов в состав ЭТС должны входить первичные электротехнические измерительные преобразователи (ЭТИП) контролируемых физиологических параметров состояния коровы и многоканальное измерительное устройство, преобразующее сигналы с выходов ЭТИП в комплексный сигнал известного формата, подлежащий дальнейшей передаче по каналу связи в устройство обработки (рисунок 11).

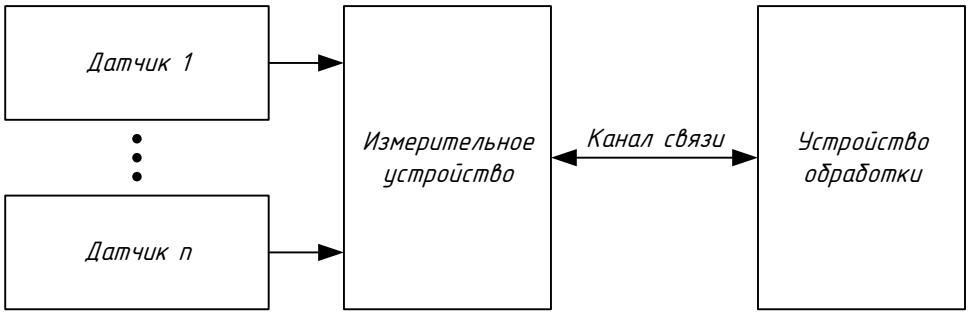


Рисунок 11 - Блок-схема электротехнического средства для идентификации предвестников родов коровы

Успешное функционирование ЭТС обусловливается адекватностью функционирования ЭТИП, конструкция которых определяется характером процессов, сопровождающих родовой акт.

Основное требование, предъявляемое к ЭТИП - однозначное соответствие комплекса выходных сигналов контролируемых физиологических параметров началу родового акта.

Требования к конструкции ЭТИП определяются тем, что считать началом родового акта, в смысле актуальности родовспоможения, и какие физические параметры ему соответствуют.

Достоверно установлены следующие предвестники родов коровы [16]:

1 - связки таза расслабляются, что приводит к западанию крестца, у корня хвоста образуется углубление. Таз становится родовым за 12-36 часов до начала выведения плода.

2 - наружные половые губы припухают и отвисают.

3 - из половой щели тянется прозрачная слизь - "поводки". "Поводки" наблюдаются за 1-2 дня до родов.

4 - молочная железа увеличивается в объеме, из сосков, при сдавливании их рукой, выделяется молозиво, обычно за 2-3 дня до родов.

5 - понижается температура тела животного на 0,4-1,2 °С за 12-50 часов до родов при ее тенденции к повышению на протяжении последнего месяца беременности.

6 - укорачивается шейка матки, что свидетельствует о начале ее раскрытия и наступлении родов в течение ближайших 2-3 часов.

При этом каждый предвестник в отдельности и даже раскрытие шейки матки не указывает однозначно на скорое наступление родов, на которое указывают только все предвестники в комплексе.

Особенности течения родов у коровы проявляются на двух стадиях. Первая стадия - это стадия раскрытия шейки матки, продолжающаяся до 12 часов, в период которой корова ведет себя достаточно спокойно, изредка продолжительно и тихо мыча. Через раскрытую шейку матки выпячивается плодный пузырь, оболочка которого обычно разрывается еще во влагалище. В момент выведения из родовых путей выступают свободные предлежащие конечности плода. Вторая стадия выведения плода длится от 20 мин. до 3-4 часов и более, в зависимости от физиологического состояния животного. В этой стадии корова обычно лежит на животе или боку, у нее наблюдается сильное общее возбуждение [16]. При появлении признаков предстоящих родов, ветврач или фермер определяют, правильно ли расположен плод, и при необходимости его поправляют. Зачастую возникает необходимость оказания родовспоможения. Обычно нормальные роды коров длятся около трех часов, а у нетелей - около четырех часов.

2.3. Физические параметры при родовом акте, пригодные для идентификации электротехническими средствами

Родовой акт коровы сопровождается физиологическими явлениями, имеющими, с некоторыми допущениями, параметры, которые могут быть автоматически идентифицированы ЭТС. В таблице 2 представлены пригодные для идентификации физические параметры во время родового акта коровы и возможные варианты их использования при построении ЭТС.

Таблица 2 - Физические параметры пригодные для идентификации ЭТС

№ п/п	Методы контроля	Размерность	Контролируемые параметры
1	Распределение температуры по поверхности тела коровы	град (г[1])	Тепловизионное распознавание образа родового акта
2	Скорость изменения температуры среды, окружающей датчик	град/сек	Мониторинг изменения температуры животного и температуры внешней среды
3	Ускорение датчика	м/сек^2	Определение ускорения выпадения датчика из влагалища
4	Частота колебаний хвоста коровы	Гц	Определение частоты колебания хвоста животного
5	Двигательная активность	шт/сек	Мониторинг отношения количества шагов животного ко времени, проводимому им лежа
6	Давление мышц влагалища	н/м^2	Определение сокращений мышц матки (схватки)
7	Время, проведенное животным в удалении от центра стада	м·сек	Определение центра стада и положения животного относительно этого центра, а так же времени, проводимого им в удалении от этого центра
8	Положение хвоста животного во время родового акта	пх^2·сек	Мониторинг положения корня хвоста животного в пространстве и времени, проведенного им в этом положении
9	Напряжение мышц живота	н/м^2	Определение сокращений мышц брюшного пресса (потуги)
10	Напряжение мышц хвоста	н/м^2	Определение мышечного напряжения корня хвоста животного во время родового акта

Анализ физических параметров при родах коров (нетелей) показывает, что одним из наиболее предпочтительных методов является измерение напряжения мышц корня хвоста. Данный метод контроля представляется достаточно технологичным с точки зрения фиксации ЭТИП непосредственно на животном, и достаточно достоверным, т.к. роды, сопровождаемые потугами, вызывают напряжения мышц животного, при этом хвост отводится в сторону. ЭТИП целесообразно закрепить у корня хвоста животного на расстоянии 200 - 300 мм от крестца.

2.4. ЭТИП регистрации мышечного напряжения корня хвоста животного.

Нами были разработаны и испытаны три типа ЭТИП регистрации мышечного напряжения корня хвоста животного, использующие разные

[1] Радиус−вектор.

[2] Положение корня хвоста, в котором хвост продолжает линию позвоночника.

физические принципы работы, но отражающие одинаковую сущность явления регистрируемого процесса:

- ЭТИП мышечного напряжения корня хвоста на основе измерения давления на чувствительный элемент;

- ЭТИП мышечного напряжения корня хвоста на основе измерения деформации чувствительного элемента;

- ЭТИП мышечного напряжения корня хвоста животного на основе измерения относительного удлинения длины окружности ТЧЭ, наполненного электропроводящим раствором.

Ниже приводится разработанные математические модели ЭТИП регистрации мышечного напряжения корня хвоста животного.

2.5. Математическая модель ЭТИП мышечного напряжения корня хвоста на основе измерения давления на чувствительный элемент.

Техническая реализация ЭТИП давления основана на применении манжеты с герметичной камерой из упругого элемента (резина). При этом камера заполняется сжатым воздухом до заданного значения. Манжета с камерой крепится к корню хвоста. Для преобразования избыточного давления в камере в электрический сигнал используется аналоговый пневмоэлектрический преобразователь.

Математическая модель ЭТИП описывается следующими физическими законами и уравнениями:

- Закон Гукка: $\vec{F}_{\text{упругости}} = -k \cdot x$;

- Закон Ньютона: $\vec{F}_{\text{действия}} = -\vec{F}_{\text{противодействия}}$;

- Закон Ома: $I = \frac{U}{R}$;

- Закон Паскаля: $P = \frac{\vec{F}}{S}$;

- Уравнение Менделеева-Клапейрона: $p \cdot V = \frac{m}{\mu} \cdot R \cdot T$;

Обобщенная схема ЭТИП давления приведена на рисунке 12.

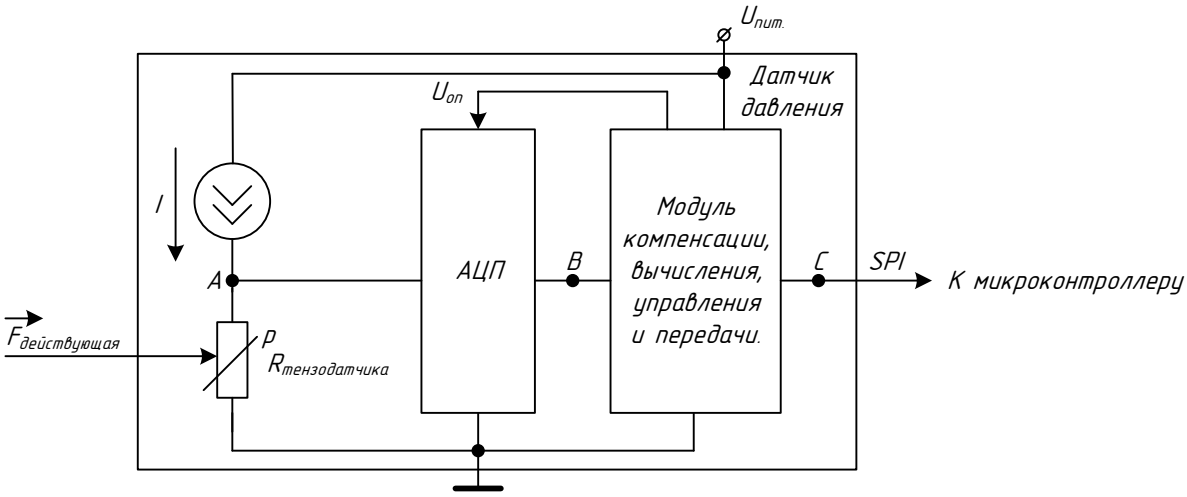


Рисунок 12 - Обобщенная функциональная схема ЭТИП давления

Основным рабочим элементом схемы является тензодатчик сопротивлением $R_{тензодатчика}$. Из теории работы тензодатчиков [152] известно следующее соотношение: $\frac{\Delta R}{R_0} = \gamma \cdot \frac{\Delta L}{L_0}$, где $\frac{\Delta L}{L_0}$ - относительная деформация тензодатчика, $\frac{\Delta R}{R_0}$ - относительное сопротивление тензодатчика, γ - коэффициент усиления тензодатчика - тензочувствительность. Для разных материалов тензочувствительность разная. К примеру, для металлов она составляет 2 - 4,5, для полупроводников - более 150.

На рисунке 12 показано, что через тензодатчик сопротивлением $R_{тензодатчика}$ проходит постоянный ток силой I. На основании закона Ома напряжение в точке "А" соответствует выражению:

$$U = I \cdot R_{тензодатчика}. \quad (1)$$

На тензодатчик сопротивлением $R_{тензодатчика}$ действует сила $\vec{F}_{действующая}$, сообщаемая рабочим телом ЭТИП. Под действием этой силы $R_{тензодатчика}$ представляется из двух составляющих: постоянной (R_0), зависящей от начального давления рабочей среды (воздуха) в рабочем теле ЭТИП, задаваемой во время его фиксации на хвосте животного, и переменной (ΔR), зависящей от мышечного тонуса хвоста. На основании этого (1) можно записать:

$$U = I \cdot (R_0 + \Delta R). \quad (2)$$

Из $\frac{\Delta R}{R_0} = \gamma \cdot \frac{\Delta L}{L_0}$ можно выразить переменную составляющую (ΔR) для тензодатчика:

$$\Delta R = \gamma \cdot \frac{R_0}{L_0} \cdot \Delta L. \quad (3)$$

Из закона Гукка выражаем длину изгиба тензодатчика под воздействием силы $\vec{F}_{\text{действия}}$:

$$\Delta L = \frac{1}{k} \cdot (-\vec{F}_{\text{упругости}}). \quad (4)$$

Сила упругости в законе Гукка соответствует силе противодействия в законе Ньютона. Соответственно, (4) можно преобразовать:

$$\Delta L = \frac{1}{k} \cdot \vec{F}_{\text{действия}}. \quad (5)$$

Подставляя (3) в (5), получаем выражение:

$$\Delta R = \frac{\gamma}{k} \cdot \frac{R_0}{L_0} \cdot \vec{F}_{\text{действия}}. \quad (6)$$

Сила, действующая на тензодатчик, выражается из закона Паскаля:

$$\vec{F}_{\text{действия}} = S \cdot P. \quad (7)$$

Подставляя (6) в (7), получаем выражение:

$$\Delta R = \frac{\gamma}{k} \cdot \frac{R_0}{L_0} \cdot S \cdot P. \quad (8)$$

Давление (P), действующее на тензодатчик, выражается из уравнения Менделеева-Клапейрона, где $R_{газ}$ - универсальная газовая постоянная, равная 8,31441 $\frac{\text{Дж}}{\text{моль}\cdot\text{К}}$:

$$P = \frac{\frac{m}{\mu} \cdot R_{\text{газ}} \cdot \text{T}}{V_{\text{рабочего тела}}}. \quad (9)$$

Подставляя (8) в (9), получаем выражение:

$$\Delta R = \frac{\gamma}{k} \cdot \frac{R_0}{L_0} \cdot S \cdot \frac{\frac{m}{\mu} \cdot R_{\text{газ}} \cdot \text{T}}{V_{\text{рабочего тела}}}. \quad (10)$$

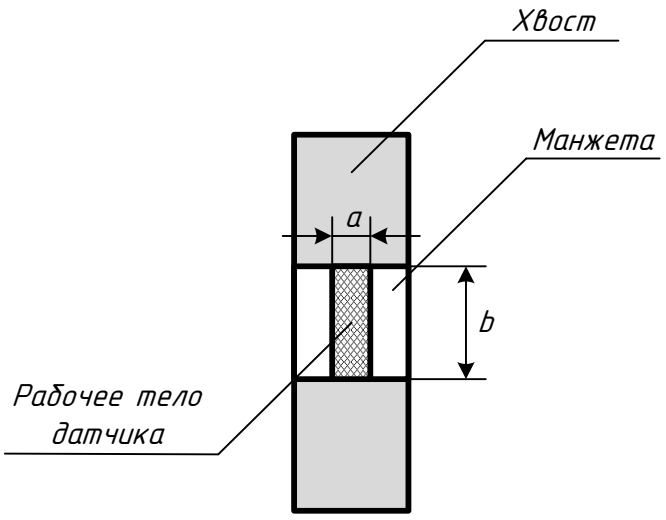


Рисунок 13 - Расположение рабочего тела ЭТИП на хвосте

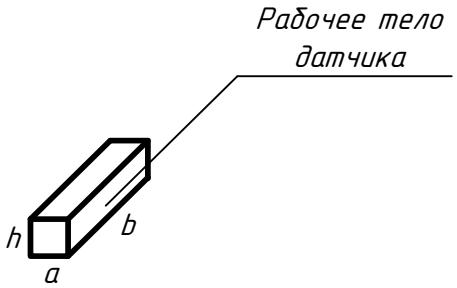


Рисунок 14 - Схема обозначения сторон рабочего тела ЭТИП

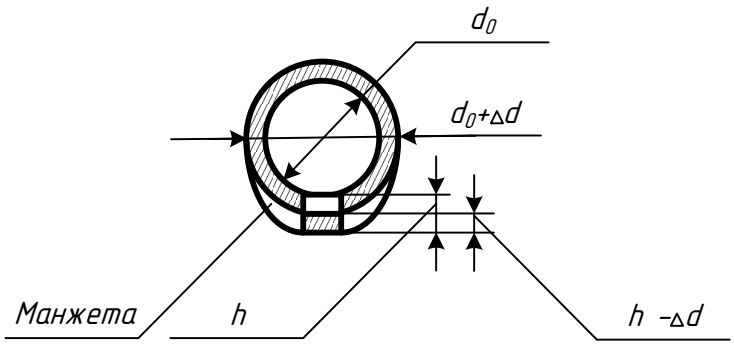


Рисунок 15 - Схема изменения диаметра хвоста животного

На рисунке 13 изображено расположение рабочего тела ЭТИП на хвосте животного. Рабочим телом ЭТИП является резиновая груша объемом *V*, наполненная воздухом, изображенная на рисунке 14. Во время явлений,

происходящих с хвостом (рисунок 15), мышечный тонус изменяется, изменяя его диаметр, что приводит к изменению объема рабочего тела $\left(V_{\text{рабочего тела}} = f\left(d_{\text{диаметр хвоста}}\right)\right)$.

Объем рабочего тела выражается:

$$V_{\text{рабочего тела}} = a \cdot b \cdot h. \tag{11}$$

На основании (11) и принципа функционирования рабочего тела ЭТИП, отраженного на рисунке 14, объем выражается:

$$V_{\text{рабочего тела}} = a \cdot b \cdot h - a \cdot b \cdot \Delta h = a \cdot b \cdot (h - \Delta h). \tag{12}$$

$\Delta h = \Delta d = d - d_0$, где $d \geq d_0$, d - диаметр хвоста. (13)

На основании (12) и (13) получается выражение объема рабочего тела ЭТИП от диаметра хвоста животного:

$$V_{\text{рабочего тела}} = a \cdot b \cdot (h - d + d_0). \tag{14}$$

Из (2), (10) и (14) следует выражение вычисления напряжения в точке "А" в зависимости от изменения диаметра хвоста:

$$U = I \cdot R_0 \cdot \left[1 + \frac{\gamma}{k} \cdot \frac{S}{L_0} \cdot \left(\frac{\frac{m}{\mu} \cdot R_{\text{газ}} \cdot T}{a \cdot b \cdot (h - d + d_0)}\right)\right], \tag{15}$$

где:

- I (А) - электрический ток, протекающий через тензодатчик;

- $R0$ (Ом) - постоянная составляющая сопротивления тензодатчика;

- γ (В/м) - тензочувствительность;

- k $\left(\frac{\text{кг}}{\text{с}^2}\right)$ - коэффициент упругости материала, из которого изготовлен тензодатчик;

- S (м^2) - площадь рабочей поверхности тензодатчика;

- L_0 (м) - постоянная составляющая длины тензодатчика;

- $R_{\text{газ}}$ $\left(\frac{\text{Дж}}{\text{моль·К}}\right)$ - универсальная газовая постоянная 8,31441 $\frac{\text{Дж}}{\text{моль·К}}$;

- μ $\left(\frac{\text{кг}}{\text{моль}}\right)$ - молярная масса рабочего вещества (воздуха) рабочего тела ТЧЭ равна $28{,}98 \cdot 10^{-3}$ $\frac{\text{кг}}{\text{моль}}$;

- m (кг) - масса рабочего вещества (воздуха) рабочего тела ТЧЭ;

- T (К) - температура окружающей среды рабочего тела ТЧЭ;

- a (м) - ширина рабочего тела ТЧЭ;

- b (м) - длина рабочего тела ТЧЭ;

- h (м) - глубина (высота) рабочего тела ТЧЭ;

- d (м) - диаметр хвоста;

- d_0 (м) - постоянная составляющая диаметра хвоста.

Цифровой код (N), получаемый в точке "B", эквивалентный значению напряжения в точке "A", выражается:

$$N = \frac{U}{U_{\text{оп}}} \cdot (2^n - 1), \tag{16}$$

где $U_{оп}$ - опорное напряжение, используемое АЦП, а n - разрядность АЦП.

В ЭТИП регистрации мышечного напряжения корня хвоста используется ЭТИП давления HSCDAND030PGSA3, измеряющий давление в "фунт-сила на квадратный дюйм" (psi). В документации на данный ЭТИП приводится выражение для вычисления цифрового кода N, соответствующего для точки "C" рисунка 12:

$$N = \frac{80\%}{P_{max} - P_{min}} \cdot (P - P_{min}) + 10\%. \tag{17}$$

Выразив из (17) величину P в паскалях, действующую на тензодатчик, получим выражение:

$$P = \left[(N - 0.1) \cdot \left(\frac{P_{max} - P_{min}}{0.8}\right) + P_{min}\right] \cdot 6894{,}76. \tag{18}$$

Подставив в (18) выражение (16), получим:

$$P = \left[\left(\frac{U}{U_{\text{оп}}} \cdot (2^n - 1) - 0.1\right) \cdot \left(\frac{P_{max} - P_{min}}{0.8}\right) + P_{min}\right] \cdot 6894{,}76. \tag{19}$$

Подставив в (19) выражение (15), получим математическую модель для выражения давления, зависящего от диаметра хвоста, действующего на тензодатчик:

$$P = \left[\left(\frac{I \cdot R_0 \cdot \left(1 + \frac{\gamma}{k} \cdot \frac{S}{L_0} \cdot \left(\frac{\frac{m}{\mu} \cdot R_{газ} \cdot T}{a \cdot b \cdot (h - d + d_0)} \right) \right)}{U_{on}} \cdot (2^{n-14} - 1) - 0.1 \right) \cdot \left(\frac{P_{max} - P_{min}}{0.8} \right) + P_{min} \right] \cdot 6894{,}76. \quad (20)$$

Математическая модель (20) позволяет получить зависимость $P = f(d)$, основываясь на которой можно построить математические модели регистрации функциональной деятельности органов промежности коровы.

2.6. Математическая модель ЭТИП мышечного напряжения корня хвоста на основе измерения деформации чувствительного элемента.

Техническая реализация ЭТИП мышечного напряжения основана на применении манжеты из упругого токопроводящего элемента (токопроводящая резина). Манжета крепится к корню хвоста. Для преобразования мышечного напряжения корня хвоста в электрический сигнал используется АЦП со схемой корректировки сигнала.

Обобщенная структурная схема ЭТИП деформации приведена на рисунке 16.

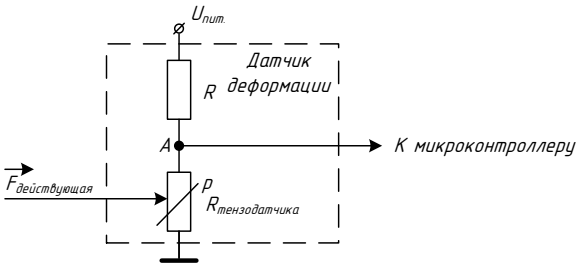


Рисунок 16 - Обобщенная структурная схема ЭТИП деформации

Основным рабочим элементом схемы является тензодатчик сопротивлением $R_{тензодатчика}$. Из теории работы тензодатчиков [152] известно следующее соотношение:

$$\frac{\Delta R}{R_0} = \gamma \cdot \frac{\Delta L}{L_0}, \quad (21)$$

где $\frac{\Delta L}{L_0}$ - относительная деформация тензодатчика, $\frac{\Delta R}{R_0}$ - относительное сопротивление тензодатчика, γ - коэффициент усиления тензодатчика - тензочувствительность.

На рисунке 16 отражено, что ЭТИП деформации построен по схеме делителя напряжения. На основании закона Ома напряжение в точке "А" соответствует выражению:

$$U = \frac{R}{R + R_{\text{тензодатчика}}} \cdot U_{\text{пит.}}. \tag{22}$$

На тензодатчик сопротивлением $R_{тензодатчика}$ действует сила $\vec{F}_{\text{действующая}}$, сообщаемая хвостом. Под действием этой силы $R_{тензодатчика}$ представляется из двух составляющих: постоянной (R_0), зависящей от начального давления хвоста на тензодатчик, задаваемого во время фиксации ЭТИП на хвосте, и переменной (ΔR), зависящей от мышечного тонуса хвоста. Сила, действующая на тензорезистор, представляет собой некоторую функцию от диаметра хвоста, изменяющегося под действием мышечного тонуса при изменении положения хвоста. Это позволяет использовать диаметр хвоста как основной параметр при построении математической модели ЭТИП. На основании этого можно выразить относительную деформацию тензодатчика через диаметр хвоста, используя формулу длины окружности:

$$C = \pi \cdot d. \tag{23}$$

Длина тензодатчика может быть выражена через длину окружности хвоста и коэффициента пропорциональности:

$$L = k \cdot C. \tag{24}$$

Подставляя выражение (24) в выражение (23), получаем выражение длины тензодатчика через диаметр хвоста:

$$L = k \cdot \pi \cdot d. \tag{25}$$

Подставив выражение (25) в выражение (21), получаем выражение, где относительное сопротивление тензодатчика выражается через диаметры:

$$\frac{\Delta R}{R_0} = \gamma \cdot \frac{d - d_0}{d_0}. \quad (26)$$

Соответственно, сопротивление тензодатчика можно представить выражением:

$$R_{\text{тензодатчика}} = R_0 \cdot \left(1 + \gamma \cdot \frac{d - d_0}{d_0}\right). \quad (27)$$

Подставляя выражение (27) в выражение (22), получаем выражение:

$$U = \frac{R}{R + R_0 \cdot \left(1 + \gamma \cdot \frac{d - d_0}{d_0}\right)} \cdot U_{\text{пит.}}. \quad (28)$$

Цифровой код (N), получаемый оцифровкой напряжения сигнала в точке "А", получается из выражения:

$$N = \frac{U}{U_{\text{оп}}} \cdot (2^n - 1), \quad (29)$$

где $U_{оп}$ - опорное напряжение, используемое АЦП, а n - разрядность АЦП. Подставляя выражение (28) в выражение (29), получаем выражение

$$N = \frac{\frac{R}{R + R_0 \cdot \left(1 + \gamma \cdot \frac{d - d_0}{d_0}\right)} \cdot U_{\text{пит.}}}{U_{\text{оп}}} \cdot (2^n - 1). \quad (30)$$

Если $R = R_0$, $U_{оп} = U_{пит.}$, то получается выражение:

$$N = \frac{2^n - 1}{2 + \gamma \cdot \frac{d - d_0}{d_0}}. \quad (31)$$

Выражение (31) является математической моделью на основе ЭТИП деформации ($N = f(d)$),

где:

- γ (В/м) - тензочувствительность;

- d (м) - диаметр хвоста;

- d_0 (м) - постоянная составляющая диаметра хвоста;

- n - разрядность АЦП;

Выражение (31) очень простое и позволяет строить модели цифрового кода напряжения в точке "А" тензодатчика в зависимости от диаметра хвоста.

2.7. Математическая модель ЭТИП мышечного напряжения корня хвоста животного на основе измерения относительного удлинения длины окружности тензочувствительного элемента, наполненного электропроводящим раствором.

Разработанный нами тензочувствительный элемент (ТЧЭ) является первичным преобразователем напряжения мышц корня хвоста в электрический сигнал. Принцип действия ТЧЭ основан на изменении электрического сопротивления наполнителя (раствор NaCl в концентрации 100 г/л) в зависимости от его растяжения. В ТЧЭ в качестве чувствительного элемента применена силиконовая трубка, наполненная электропроводящим раствором, к концам которой прикреплены концы манжеты из малорастяжимого материала (нейлон). Конструкция ТЧЭ представлена на рисунке 17.

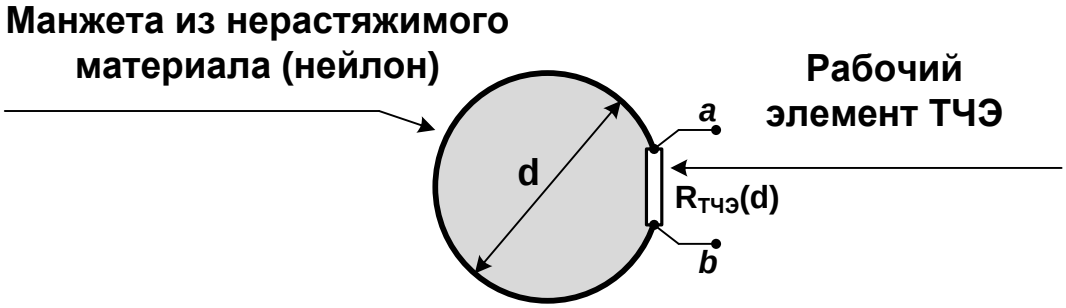


Рисунок 17 – Конструкция ТЧЭ

Электрическое сопротивление ТЧЭ равно:

$$R = \frac{\rho \cdot l}{S}. \tag{32}$$

Конструкция ТЧЭ представляет собой цилиндр, заполненный электролитом. Из этого следует, что площадь сечения (S) равна:

$$S = \frac{V}{l}. \tag{33}$$

Подставляя (32) в (33), получаем выражение:

$$R = \frac{\rho}{V} \cdot l^2, \tag{34}$$

где l представляется длиной окружности хвоста животного, которую можно выразить через его диаметр:

$$l = \pi \cdot d. \tag{35}$$

Подставляя (34) в (35), получаем выражение электрического сопротивления ТЧЭ в зависимости от диаметра:

$$R_{ТЧЭ}(d) = \frac{\rho_{раствора}}{V_0} \cdot (\pi \cdot d)^2, \tag{36}$$

где $\rho_{раствора}$ – удельное сопротивление (константа); V_0 – объем силиконовой трубки (константа); d – диаметр хвоста животного в плоскости крепления ТЧЭ.

Схема включения рабочего элемента ТЧЭ представлена на рисунке 18.

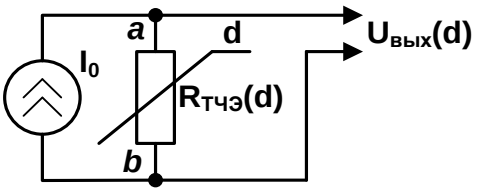


Рисунок 18 – Схема включения рабочего элемента ТЧЭ

Напряжение на выходе рабочего элемента ТЧЭ ($U_{Вых}$) равно:

$$U_{Вых}(d) = I_0 \cdot R_{ТЧЭ}(d). \tag{37}$$

Подставив (36) в (37), получим выражение $U_{Вых}$ от d для рабочего элемента ТЧЭ:

$$U_{Вых}(d) = \frac{I_0 \cdot \rho_{резины} \cdot \pi^2}{V_0} \cdot d^2. \tag{38}$$

Выражение (38) можно записать так:

$$U_{Вых}(d) = k \cdot d^2, \tag{39}$$

где:

- $k = \frac{I_0 \cdot \rho_{раствора} \cdot \pi^2}{V_0}$ - чувствительность ТЧЭ, величина постоянная;

- d – диаметр хвоста животного;

Напряжение мышц вызывает увеличение диаметра хвоста в плоскости крепления ТЧЭ, что приводит к такому же увеличению длины рабочего тела ТЧЭ. Расслабление мышц вызывает уменьшение диаметра хвоста в плоскости крепления ТЧЭ, при этом уменьшается и длина рабочего тела ТЧЭ. Электрическое сопротивление ТЧЭ возрастает пропорционально увеличению длины рабочего тела. В результате изменяется напряжения на выходе ТЧЭ, что соответствует напряжению мышц корня хвоста.

2.8. Алгоритм обработки сигналов ЭТИП мышечного напряжения для идентификации начала родов у коров

Сигнал с ТЧЭ, пропорциональный напряжению мышц корня хвоста животного, поступает на вход АЦП (аналого-цифровой преобразователь) МК, преобразующего с заданной частотой аналоговый сигнал в цифровой поток данных, который через интерфейс и устройство сопряжения поступает в программу интерпретации показаний ЭТИП, установленной на персональной электронно-вычислительной машине (ПЭВМ).

Критерии идентификации сигналов коров методом регистрации мышечного напряжения корня хвоста формируются на основе контроля параметров: $t_д$ - продолжительность рабочего сигнала (импульса) от ЭТИП, возникающего при поднятии хвоста, с.; N - количество импульсов, шт.; $t_и$ - контролируемый временной интервал, мин.; $t_п$ - временной интервал, характеризующий патологические роды, мин.

Сигнал – начало родов. Критерии идентификации:

$$\begin{cases} t_{\text{д}} > 600 \text{ с.} \\ N = 1 \\ t_{\text{п}} > 180 \text{ мин.} \end{cases} \tag{40}$$

Сигнал – дефекация или уринация. Критерии идентификации:

$$\begin{cases} t_д > 5\ с. \\ N = 1. \end{cases} \tag{41}$$

Сигнал – стресс, возникающий, например, перед или вовремя доения, кормления, взвешивания или сортировки коров. Критерии идентификации:

$$\begin{cases} t_{\partial} > 5\ с. \\ t_{и} > 60\ мин. \\ N > 3. \end{cases} \quad (42)$$

Сигнал – угроза насекомых, возникающий, например, при отпугивании хвостом слепней или оводов. Критерии идентификации:

$$\begin{cases} t_{\partial} = 1 \dots 5\ с. \\ t_{и} > 10\ мин. \\ N = 3 \dots 10. \end{cases} \quad (43)$$

Указанные критерии не являются фиксированными значениями и могут легко корректироваться программными средствами с учетом индивидуальных особенностей животного. Например, это может быть вызвано породными различиями или известной предрасположенностью конкретной коровы к трудным родам.

Программа интерпретирует поток данных с ЭТИП и с учетом математических моделей физиологических явлений, заложенных в нее, с определенной достоверностью сообщает оператору об отеле у коровы. Оператор, опираясь на полученные сведения, после их перепроверки принимает те или иные управляющие решения.

В качестве типовых явлений, происходящих с хвостом животного, нами были выбраны явления, связанные с уринацией и дефекацией, так как: 1 – присутствуют у всех животных вне зависимости от пола и возраста; 2 – их периодичность намного более высокая, чем у явлений, происходящих с хвостом больного или беременного животного; 3 – за счет своей простоты для наблюдения, регистрации и интерпретации от разработчика не требуется квалификации в области ветеринарии. Перечисленные обстоятельства позволяют разработать и реализовать относительно простую математическую модель детектирования таких явлений. Исключение продетектированных явлений уринации и дефекации из общего потока явлений позволяет интерпретировать оставшиеся явления как явления в мочеполовой, выделительной и пищеварительной системе, требующие акцентирования внимания со стороны персонала.

В процессе работы ЭТИП мы располагаем только полученными ранее и текущим оцифрованным значениями сигнала. В этом случае целесообразно использовать причинный фильтр, где время - независимая переменная, и реакция фильтра возможна только на прошедшие значения (причины). Для компенсации фазового сдвига нами были использованы фильтры с конечной импульсной характеристикой (КИХ-фильтр), описываемые общим выражением:

$$y_n = \sum_{k=-\infty}^{\infty} c_k u_{n-k},$$

где u_n - входные данные, $c_к$ - постоянные фильтра, y_n - выходные сигналы.

Коэффициенты фильтра $c_к$ предполагаются постоянными в течение большей части времени, соответственно, фильтр является инвариантным во времени.

Алгоритм (математическая модель) детектирования физиологических явлений построен следующим образом: поток данных с ЭТИП подается на первое звено (44) и (45) полосового цифрового фильтра второго порядка.

$$H0(i) = \frac{\sum_i H_i}{S}, \tag{44}$$

$$H1(i) = \frac{\sum_i H_i}{M}, \tag{45}$$

где:

- H – поток оцифрованных данных с ЭТИП;

- S – сглаживающая весовая функция по n точкам;

- M – сглаживающая весовая функция по n/2 точкам;

Полученные значения *H0* и *H1* подаются на второе звено (46) и (47) полосового цифрового фильтра второго порядка.

$$H2(i) = \frac{\sum_i H0_i}{S}, \tag{46}$$

$$H3(i) = \frac{\sum_i H1_i}{M}. \tag{47}$$

Фильтром второго порядка отфильтровывается высокочастотная составляющая помехи во входном сигнале с ЭТИП.

Очищенный от высокочастотной составляющей помехи сигнал (46) и (47) поступает на дифференциальное звено цифрового фильтра (48),

отфильтровывающее низкочастотную составляющую помехи во входном сигнале с ЭТИП.

$$H4(i) = (H3_i - H2_i). \tag{48}$$

Очищенный от высокочастотной и низкочастотной составляющей помехи сигнал (48) поступает на вход усилителя полезного сигнала (49).

$$H5(i) = |H4_i \cdot 100|^4. \tag{49}$$

Полученные значения (49) поступают в модуль распознавания образа явления. Принцип работы по распознаванию образа сводится к сравнению гистограммы текущего явления с эталонными гистограммами явлений. Для этого используется критерий Хи-квадрат для сравнения двух гистограмм:

$$H6 = \sum_i \frac{(H5(i) - Hэ(i))^2}{H5(i) + Hэ(i)}, \tag{50}$$

где:

- $H_э$ – поток эталонных данных;

Результатом работы является значение *H6* в диапазоне от 0 до ∞, где 0 – полное совпадение гистограмм, а значит и полное совпадение детектированного явления с эталонным явлением, а ∞ - их полное несовпадение. Из этого следует, что, редактируя пороговый уровень *H6* для конкретной эталонной гистограммы, можно корректировать чувствительность системы на конкретное явление.

Работу построенного фильтра проверяем реакцией на функцию Хевисайда (51), рисунок 19.

$$\theta(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}. \tag{51}$$

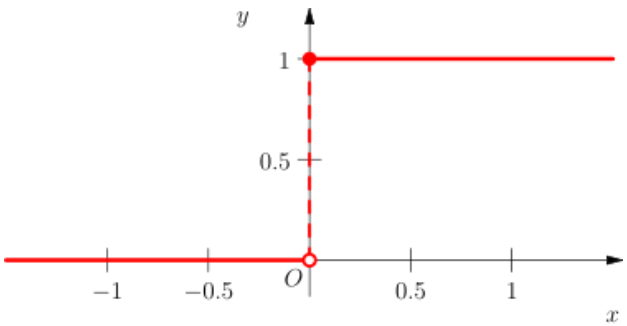


Рисунок 19 – Графическое представление функции Хевисайда (51)

Результат работы фильтра на функцию (51) приведены на рисунке 20.

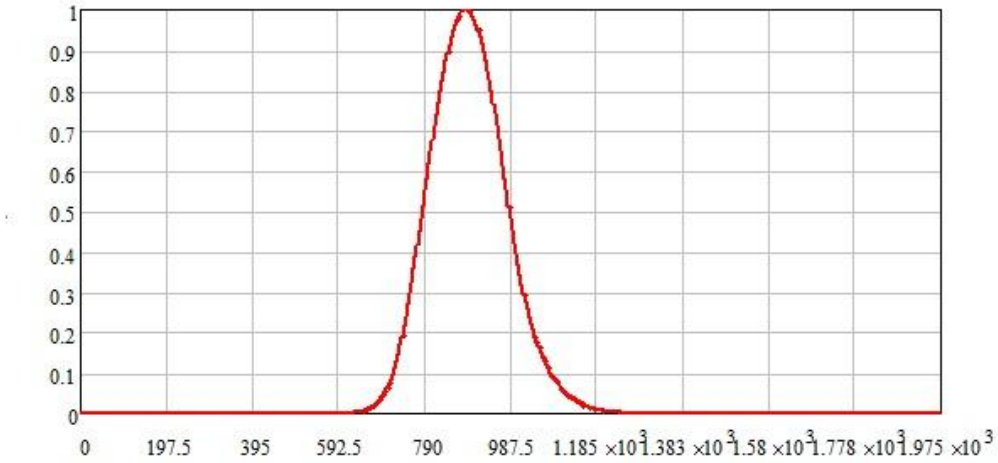


Рисунок 20 – Графическое представление реакции СЦФ (50) на функции Хевисайда (51)

Из рисунка 20 видно, что разработанный СЦФ (50) пригоден для работы. Он устойчив на единичное воздействие – отсутствует затухающий переходный процесс, а само событие перехода из одного состояния в другое резко выражено.

Результатом работы алгоритма является значение в диапазоне от 0 до ∞, где 0 – полное совпадение детектируемого явления с заложенной моделью явления, а ∞ - их полное несовпадение.

ГЛАВА 3. ПРОГРАММА, МЕТОДИКА И ЭКСПЕРИМЕНТАЛЬНАЯ УСТАНОВКА ДЛЯ ПРОВЕДЕНИЯ ИССЛЕДОВАНИЙ.

С целью проверки разработанных математических моделей и возможности реализации ЭТС для определения отела у коров нами был разработан и изготовлен: 1 – прототип системы (далее «Прототип»), представленный на рисунке 21. Отличие прототипа от ЭТИП заключается в передаче данных с ЭТИП в ПЭВМ не по радиоканалу, а с помощью проводного интерфейса RS-485, что на начальном этапе позволило ускорить процесс разработки текущего варианта ЭТС; 2 – модель хвоста животного; 3 – стенд для проведения исследования характеристик ТЧЭ (рисунок 23). 4 − разработано специализированное программное обеспечение (СПО) для ПЭВМ. Проведены четыре класса экспериментов: 1 – на модели хвоста животного; 2 – на руке человека; 3 – на хвосте животного; 4 - при родах животного. Экспериментальные исследования проведены на молочной ферме зоостанции РГАУ–МСХА имени К.А. Тимирязева и фермы ЗАО «Агрофирма «Бунятино».

Состав стенда: 1 – прототип; 2 – ПЭВМ с СПО; 3 – модель хвоста животного; 4 – воздушный компрессор; 5 – манометр «МТИ»; 6 – цифровой осциллограф; 7 – цифровой мультиметр.

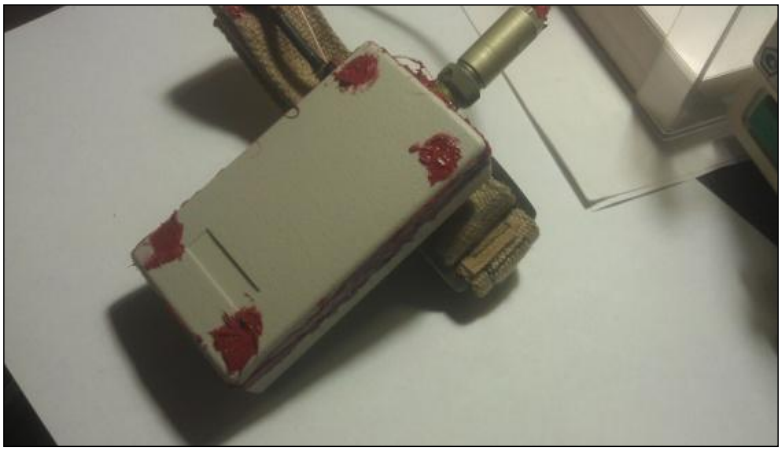


Рисунок 21 – Прототип ЭТИП ЭТС для определения отела у коров

Рисунок 22 – Стенд для проведения исследования прототипа на модели хвоста животного

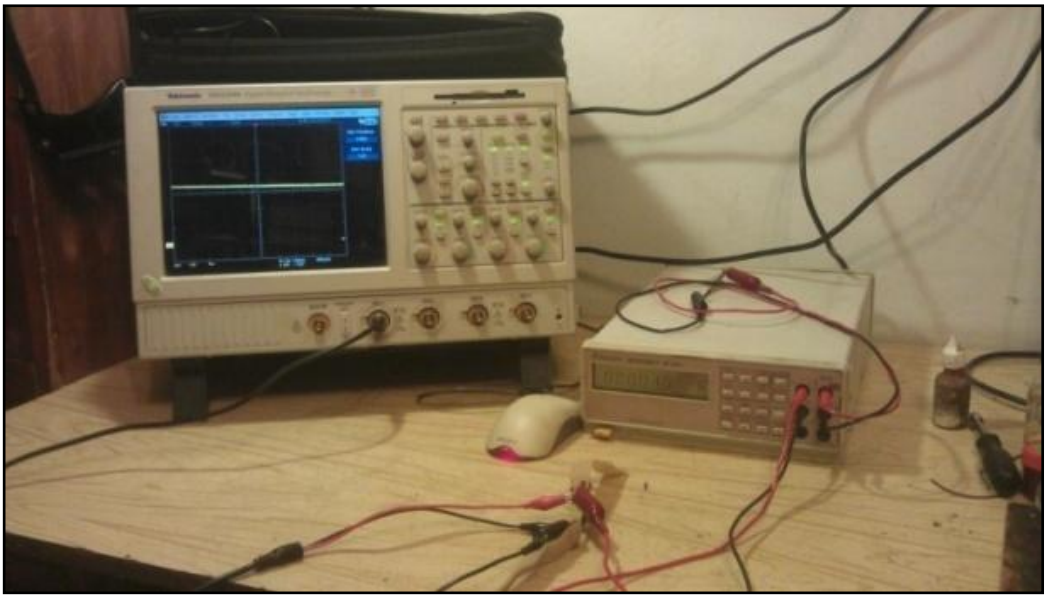


Рисунок 23 – Стенд для исследования характеристик ТЧЭ

3.1. Методика исследований ЭТС на модели хвоста животного.

Целью исследований является определение статической характеристики ТЧЭ при радиальном воздействии силы растяжения и установлении рабочего диапазона ТЧЭ по полученной статической характеристике.

Детектируемые явления с хвостом животного происходят вне зависимости от положения в пространстве тела животного. В любом случае, положение хвоста относительно тела животного связано с напряжением мышц корня хвоста.

Рисунок 24 – Модель хвоста коровы с закрепленным прототипом

Модель хвоста животного представляет собой резиновую камеру, расширение которой под давлением воздуха из компрессора происходит в узкой области, на которой закреплен ЭТИП (рисунок 24). Диаметр камеры соответствует усредненному диаметру хвоста взрослого животного. Имитация тонуса мышц корня хвоста производилась посредством изменения давления воздуха в камере, что приводило к изменению диаметра камеры и реакции ТЧЭ.

3.2. Методика исследований ЭТС при измерении мышечного напряжения на руке человека.

Целью исследований является получение динамических характеристик явлений напряжения и расслабления мышц, характеризующихся соответствующими неизменными формами сигнала получаемого с ТЧЭ, что позволит сделать вывод о возможности детектирования мышечного напряжения хвоста животного.

ТЧЭ закреплялся на руке человека, имитирующей мышечное напряжение корня хвоста животного (рисунок 25), и включался в электрическую цепь по схеме с рисунка 18. Через ТЧЭ пропускался постоянный ток I_0 = 0,1 мА. К контактам a и b, для регистрации данных, подключался запоминающий осциллограф. Для повышения качества лабораторного исследования параллельно с ТЧЭ подключался конденсатор емкостью 1 мкФ, подавляющий

высокочастотные помехи и наводки от близкорасположенного электрооборудования.

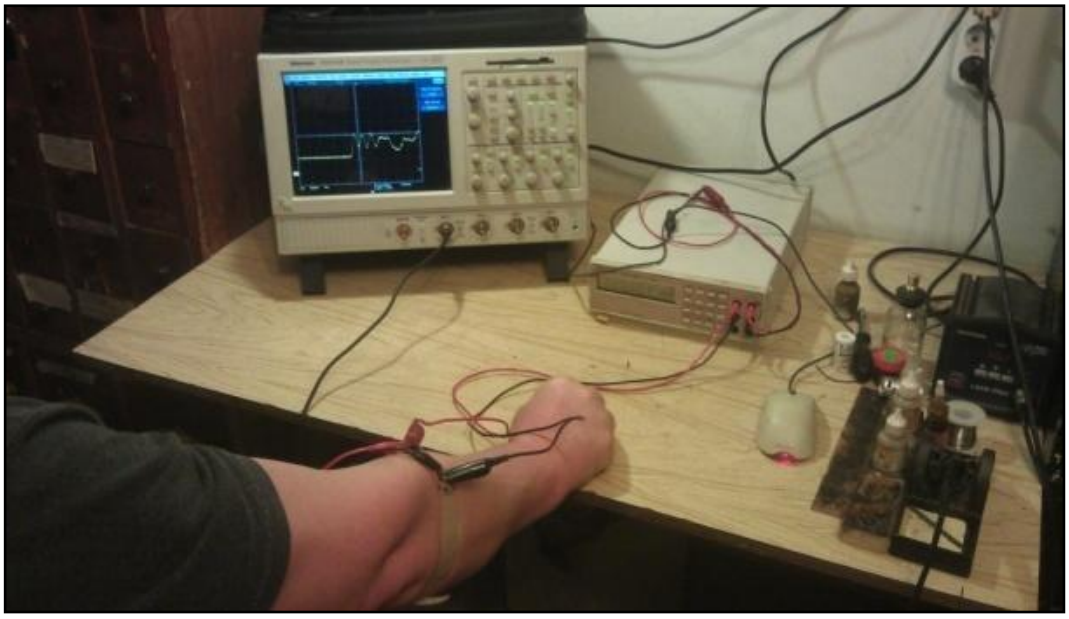


Рисунок 25 – Рука человека, имитирующая мышечное напряжение корня хвоста животного, с закрепленным на ней ТЧЭ ЭТИП

3.3. Методика исследований ЭТС при измерении мышечного напряжения на хвосте животного.

Целью исследований является проверка работы разработанных ЭТИП и математической модели детектирования явлений, происходящих с хвостом животного, и получение базового материала для построения математической модели детектирования явления родов у коровы. Исследования проводились на десяти коровах черно-пестрой породы.

ЭТИП (рисунок 28) закреплялся на животном (рисунок 26). Он является автономным устройством, т.к. питается от аккумуляторной батареи, расположенной в корпусе.

Рисунок 26 - ЭТИП и ТЧЭ, закрепленные на животном (выделено зеленым)

При проведении эксперимента осуществлялась видеозапись, синхронизированная с сигналом ТЧЭ. С помощью специального приспособления на хвосте исследуемого животного прикреплялся ЭТИП, регистрирующий мышечные напряжения корня хвоста в реальном масштабе времени и передающий актуальные данные в ПЭВМ.

Запись видеопотока и потока данных осуществлялась с незначительными задержками, вызванными реакцией оборудования, записывающего информацию.

3.4. Методика исследований ЭТС при родах животного.

Целью исследований является проверка работы разработанных ЭТИП и математической модели детектирования явления отела коровы.

Исследования проводились на молочной ферме зоостанции РГАУ-МСХА имени К.А. Тимирязева в 2015 г.

Автономно работающий ЭТИП (рисунок 27) закреплялся на готовящемся к родам животном с помощью специального приспособления. При проведении данного исследования ЭТИП закреплялся на животном за пять дней до предполагаемого отела.

Рисунок 27 - ЭТИП (выделено зрасным) и ТЧЭ (выделено зеленым), закрепленные на рожающем животном

При проведении эксперимента осуществлялась видеозапись и регистрация сигнала ТЧЭ с передачей данных на ПЭВМ.

Запись видеопотока и потока данных осуществлялась с незначительными задержками, вызванными реакцией регистрирующего оборудования.

ГЛАВА 4. ЭЛЕКТРОТЕХНИЧЕСКИЕ СРЕДСТВА ДЛЯ МОНИТОРИНГА НАЧАЛА ОТЕЛА У КОРОВ.

ЭТС включает следующие составные части (рисунок 28): 1 – устройство ЭТИП, 2 – устройство сопряжения с ПЭВМ, 3 – специальное программное обеспечение для интерпретации полученных данных. ЭТС интегрирована с системами сотовой связи для передачи фермеру SMS-сообщений о критических событиях, происходящих с животными в режиме реального времени.

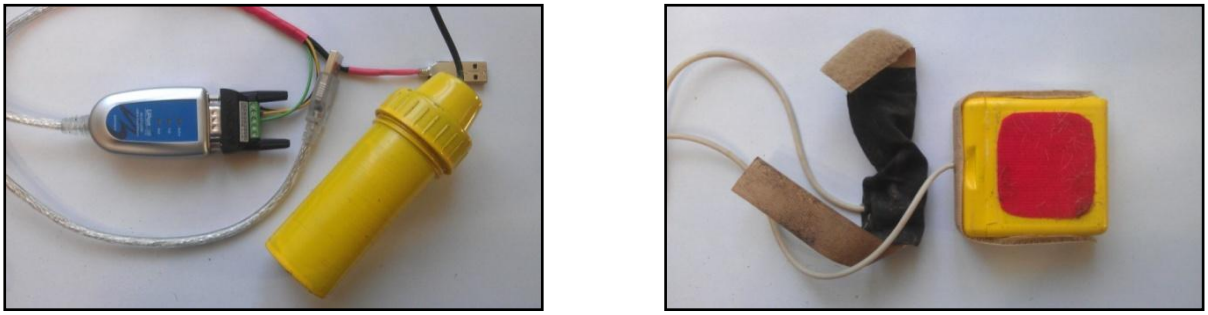


Рисунок 28 – ЭТС для определения отела у коров. Слева - устройство сопряжения с ПЭВМ. Справа – устройство ЭТИП

В таблице 3 приведены технические характеристики ЭТС.

Таблица 3 – Технические характеристики ЭТС

№ п/п	Параметр	Значение	Примечание
1	Время непрерывной работы, дни.	70	Устройство ЭТИП. При использовании аккумуляторной батареи типа 18650 (6000 мА/ч).
2	Количество одновременно обслуживаемых ЭТИП, шт.	До 6 включительно	Устройство сопряжения.
3	Дальность действия, м	20	Устройство ЭТИП и устройство сопряжения.
4	Зона покрытия, м	Зона покрытия сетей сотовой связи	
5	Рабочий диапазон частот, ГГц.	2.4	Нелицензируемый диапазон частот ISM.
6	Напряжения питания, В.	3.3	Устройство ЭТИП.
7	Габариты (ШхВхГ), мм.	73x73x30	Устройство ЭТИП.
8	Вес, г.	100	Устройство ЭТИП.

ЭТС имеет модульную структуру (рисунок 29) и разделяется на две основные части: 1 – устройство ЭТИП; 2 – система принятия, доставки и обработки полученных данных. Такая структура позволяет получить универсальный комплекс, способный контролировать различные физиологические параметры от

разных животных и выдавать оповещающие сообщения фермеру для своевременного оказания требуемой помощи животным.

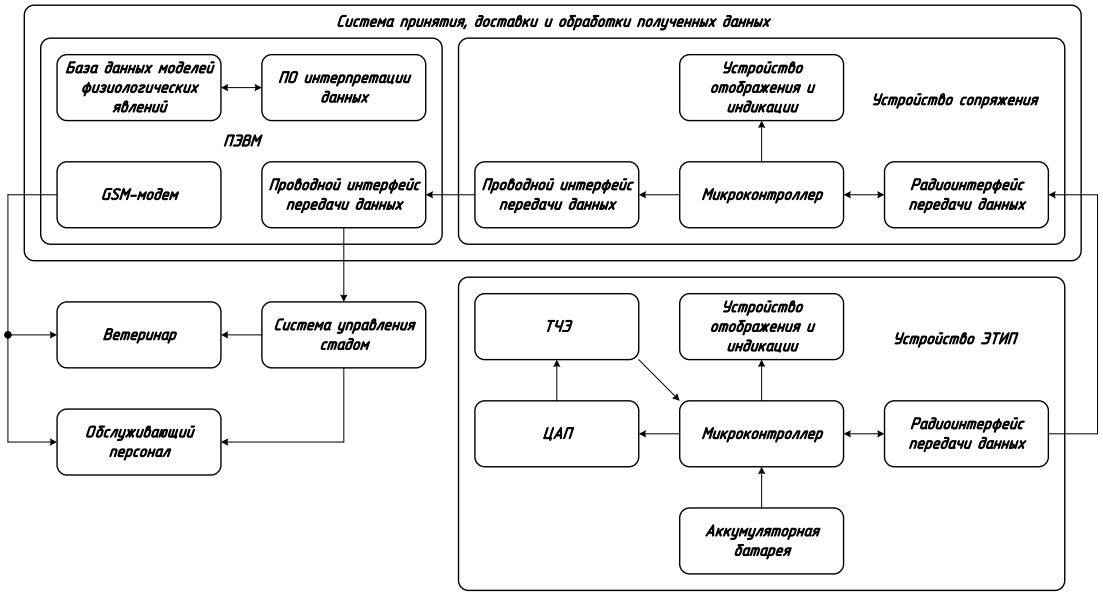


Рисунок 29 – Структура электротехнических средств для мониторинга начала отела у коров

Устройство ЭТИП закрепляется на животном. Осуществляет регистрацию физиологических явлений, происходящих с животным, и преобразование этих явлений в форму, удобную для дальнейшей обработки. Оно соединяется с устройством сопряжения с помощью радиоинтерфейса передачи данных. Питается от аккумуляторной батареи, расположенной в корпусе устройства. Расчетное время автономной работы зависит от емкости аккумуляторной батареи и составляет не менее семи дней. В устройство ЭТИП монтируются первичные преобразователи данных. Их типовой и количественный состав зависит от вида физиологической информации, регистрируемой ЭТИП. Первичными преобразователями могут быть: МЭМС - модуль акселерометра, магнитометра, гироскопа, датчик температуры, влажности, ТЧЭ и д.р. Внешний вид устройства ЭТИП приведен на рисунке 30.

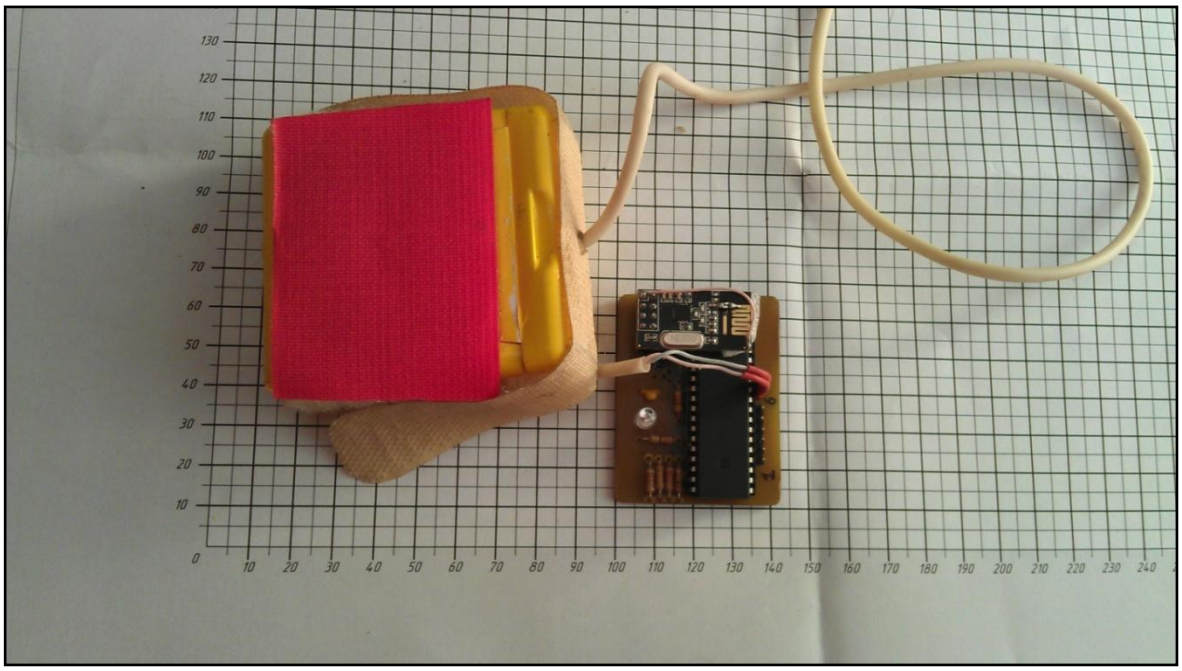


Рисунок 30 – Внешний вид внутреннего устройства ЭТИП

Устройство сопряжения с ПЭВМ подключается к ПЭВМ и располагается в непосредственной близости от нее. Осуществляет коммуникацию между ПЭВМ и устройством ЭТИП. Может взаимодействовать с несколькими устройствами ЭТИП одновременно. Внешний вид устройства сопряжения с ПЭВМ приведен на рисунке 31.

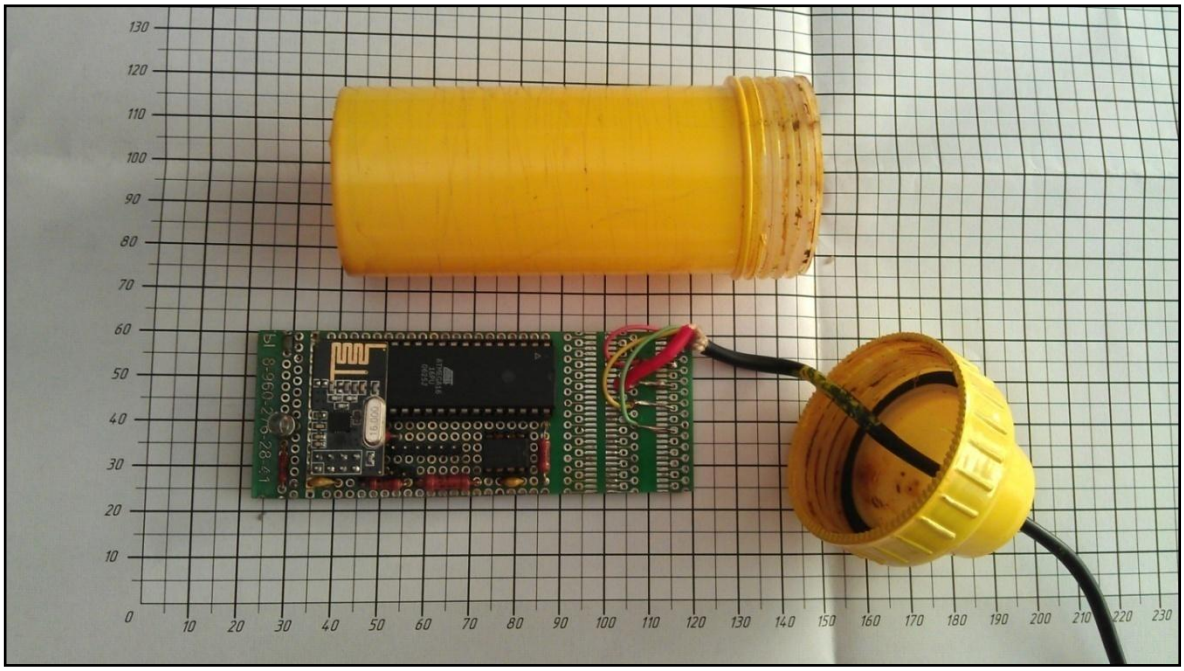


Рисунок 31 – Внешний вид внутреннего устройства блока устройства сопряжения с ПЭВМ

ПЭВМ со специализированным ПО является основным центром обработки информации, полученной с ЭТИП. На ней установлено СПО интерпретации полученных данных. СПО подключено к базе данных моделей физиологических явлений, являющейся неотъемлемой его частью. База данных моделей физиологических явлений составляется квалифицированными ветеринарами. ПО имеет возможность интеграции с системой управления стадом.

К ПЭВМ подключается стандартный GSM-модем (рисунок 32), с помощью которого СПО может удаленно оповестить фермера о наступлении заданного события через SMS-сообщение. Такими событиями являются: 1 – начало родов; 2 – половая охота; 3 – стресс. В сообщении передаются следующие параметры: 1 – наименование события; 2 - идентификационный номер животного; 3 - время события. Подключение модема к ПЭВМ осуществляется по стандартному последовательному порту. Управление модемом и передача данных ведется с использованием стандартных команд AT.

Рисунок 32 – Пример стандартного GSM-модема, подключаемого к ПЭВМ для передачи SMS- и/или MMS-сообщения

4.1. Аппаратная часть.

4.1.1. Устройство ЭТИП.

4.1.1.1. Схема электрическая структурная.

Схема электрическая структурная устройства (рисунок 33) содержит следующие функциональные модули:

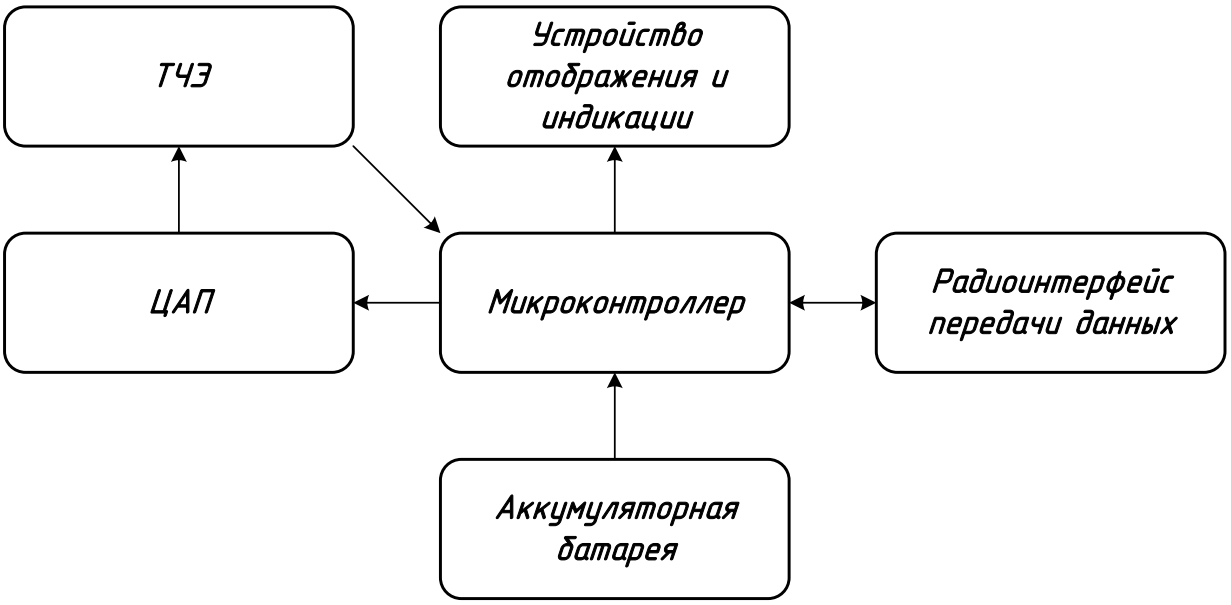


Рисунок 33 – Устройство ЭТИП. Схема электрическая структурная

- Аккумуляторная батарея;
- Микроконтроллер с ядром AVR;
- Радиоинтерфейс передачи данных;
- ТЧЭ;
- Устройство отображения и индикации;
- ЦАП.

Аккумуляторная батарея является устройством, обеспечивающим автономное электропитание устройства ЭТИП. В изделии применена аккумуляторная батарея типа 18650 (Литий-ионная, 3,7 В, 6000 мА/ч).

Микроконтроллер является основным управляющим блоком схемы. Он управляет логикой получения и обработки данных с ТЧЭ, инициализацией радиоинтерфейса передачи данных, и осуществляет передачу полученных данных с ТЧЭ.

Радиоинтерфейс передачи данных осуществляет передачу полученной от микроконтроллера информации по радиоканалу на частоте 2,4 ГГц в нелицензируемом диапазоне частот ISM устройству сопряжения с ПЭВМ.

ТЧЭ является первичным преобразователем напряжения мышц корня хвоста в аналоговый электрический сигнал.

Устройство отображения и индикации является интерфейсом "человек-машина" и осуществляет отображение внутреннего состояния изделия способом, пригодным для непосредственного восприятия человеком.

ЦАП является вспомогательным блоком связующим ТЧЭ с МК. Он обеспечивает стабильность тока (I_0) через рабочий орган ТЧЭ. Это является одним из слагаемых получения достоверных показаний о напряжении мышц корня хвоста.

Устройство ЭТИП обеспечивает:

- Преобразование напряжений мышц корня хвоста в аналоговый электрический сигнал;
- Передача значений оцифрованного аналогового электрического сигнала устройству сопряжения с ПЭВМ;
- Индикация внутреннего состояния изделия во время работы и при внештатных ситуациях.

4.1.1.2. Обоснование выбора элементной базы.

Схема электрическая принципиальная (рисунок 35) построена с использованием следующих СБИС:

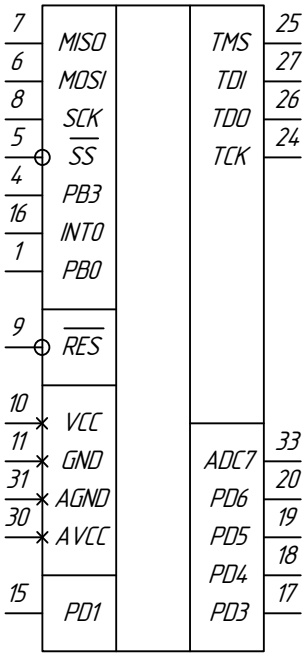


Рисунок 34 – Условно-графическое обозначение МК ATmega16L-8MU

Отличительные особенности:

- 8-разрядное ядро;
- 32x8 рабочих РОН;
- Энергонезависимые памяти программ (износостойкость: 10 тысяч циклов стирания/записи) и данных;
- 16 Кбайт внутрисхемно-программируемой флэш-памяти программ;
- 1 Кбайт статического ОЗУ памяти данных;
- SPI интерфейс;
- Внутренний осциллятор до 8 МГц;
- Рабочие напряжения: 2,7 – 5,5 В;
- Рабочая температура: -40˚С...+85˚С.

Все узлы изделия выполнены с применением стандартных СБИС, что обеспечивает малые габариты, энергопотребление, простоту и низкую стоимость изготовления, высокую надежность и, как следствие, большой период наработки изделия на отказ.

Рассмотрим возможность модификации микропрограммного обеспечения изделия. Применение распространенной и доступной элементной базы обеспечивает простоту реализации изделия. Все элементы согласуются друг с другом без дополнительных устройств.

Вышеуказанные свойства элементной базы обусловили ее выбор для разработки и изготовления изделия.

4.1.1.3. Схема электрическая принципиальная.

Рассмотрим работу изделия по схеме электрической принципиальной (рисунок 35) в положительной логике (U_{0max} = 0,7 В, U_{1min} = 4,2 В).

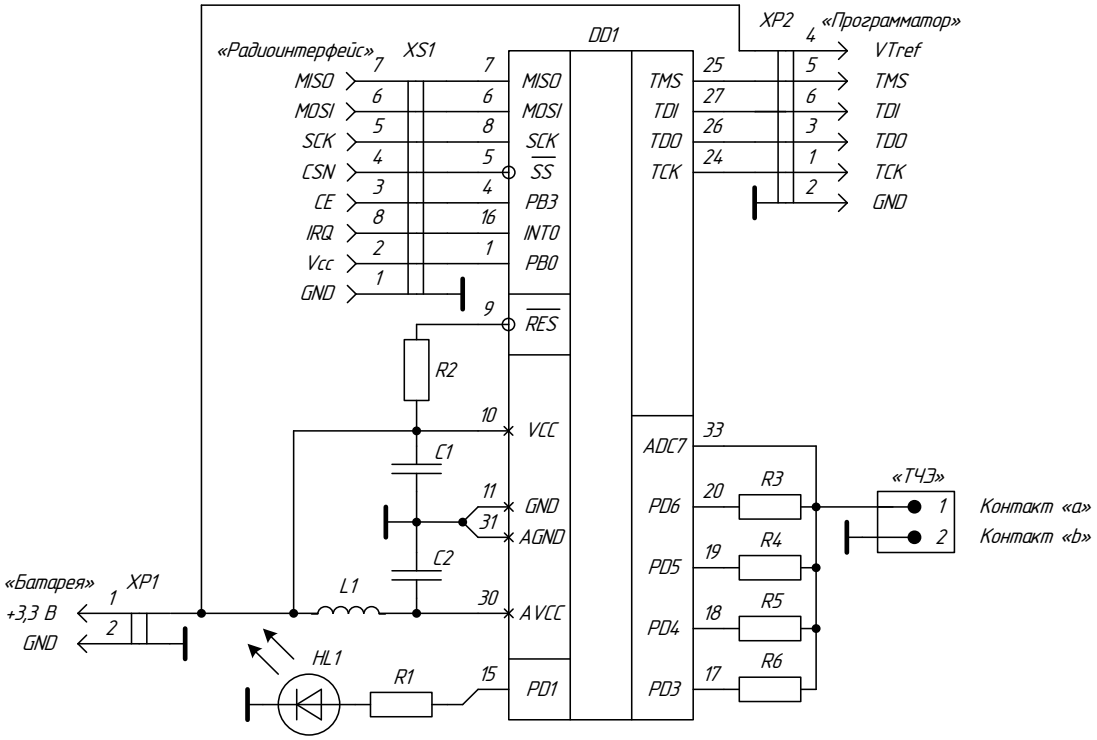


Рисунок 35 – Устройство ЭТИП. Схема электрическая принципиальная

При подачи питания +3,3 В на разъем XP1 (контакт 1) МК (DD1), с начальной задержкой в 65 мс, обусловленной стабилизацией внутреннего тактового генератора, начинает выполнение заложенной в него микропрограммы.

Микропрограмма инициализирует модуль радиоинтерфейса передачи данных nRF24L01, подключенный к разъему XS1 по интерфейсу SPI (выводы MISO, MOSI, SCK, CSN). С помощью ЦАП (резисторы R3-R6) микропрограмма задает ток I_0 через рабочий орган ТЧЭ, одновременно оцифровывая полученный с ТЧЭ сигнал (вывод ADC7). Приведенная схема совместного включения ЦАП и АЦП позволяет реализовать концепцию обратной связи для удержания сигнала с ТЧЭ в рабочем диапазоне АЦП МК.

После включения и инициализации радиоинтерфейса nRF24L01, ТЧЭ, ЦАП и АЦП МК (DD1) включает на 2 сек. зеленый светодиод (HL1), индицируя о корректной работе изделия.

Далее с частотой 16 Гц, пока не будет снято электропитание с контакта 1 разъема XP1, МК (DD1) передает оцифрованные данные с ТЧЭ по радиоинтерфейсу устройству сопряжения с ПЭВМ.

Мигание светодиода (HL1), отличное от приведенного выше, означает ошибку в микропрограмме МК (DD1).

Разъем XP2 предназначен для подключения программатора JTAG ICE MK2 к изделию и программирования МК (DD1) через промышленный интерфейс программирования и отладки JTAG (выводы TMS, TDI, TDO, TCK).

4.1.1.4. Расчет надежности.

Расчет надежности изделия выполнен с использованием программного комплекса Автоматизированная система обеспечения надежности и качества аппаратуры (АСОНИКА-К), разработанного Московским государственным институтом электроники и математики (технический университет). АСОНИКА-К позволяет получить суммарную интенсивность отказов изделий при эксплуатации и хранении по справочным данным интенсивности отказов ЭРИ ПО АСОНИКА-К, государственным военным стандартам «Мороз-6» и «Надежность ВТ».

Надежностью называется свойство изделия выполнять заданные функции, сохраняя во времени значения установленных эксплуатационных показателей в заданных пределах, соответствующих заданным режимам и условиям использования, хранения и транспортировки.

Отказ – нарушение работоспособности изделия, без устранения которого невозможно дальнейшее выполнение функций изделия.

Безотказность – свойство изделия непрерывно сохранять работоспособность в течении некоторого времени или некоторой наработки.

Вероятность безотказной работы – вероятность, что в заданном интервале времени или в пределах заданной наработки нет отказа изделия.

При расчете инженерной надежности изделия используются следующие исходные данные схемы электрической принципиальной (рисунок 35):

Таблица 4 – Интенсивность отказов ЭРИ изделия

№ п/п	Позиционное обозначение	Наименование ЭРИ	Интенсивность отказов эксплуатационная. $\lambda_э$ [1/час]	Интенсивность отказов в режиме ожидания. $\lambda_о$ [1/час]
1	DD1	ATMega16L-8MU	$2{,}07\cdot10^{-8}$	$5{,}26\cdot10^{-10}$
2	L1	CM322522-100KL	$4{,}70\cdot10^{-10}$	$5{,}00\cdot10^{-10}$
3	C1, C2	GCM155R71C104KA55D	$3{,}18\cdot10^{-8}$	$3{,}18\cdot10^{-10}$
4	XS1	PBD-20	$1{,}60\cdot10^{-7}$	$2{,}04\cdot10^{-9}$
5	HL1	KPHHS-1005CGCK	$1{,}14\cdot10^{-9}$	$2{,}85\cdot10^{-11}$
6	R1	RC0402JR-0722R	$1{,}05\cdot10^{-8}$	$1{,}05\cdot10^{-10}$
7	R2 – R6	RG1005P		
8	XP1, XP2	M50-3600542	$1{,}84\cdot10^{-7}$	$4{,}61\cdot10^{-10}$

Общая интенсивность отказа ЭРИ равна $6{,}77291092094366\cdot10^{-7}$. Общая интенсивность отказа в режиме ожидания $5{,}2821732641302\cdot10^{-9}$.

Графическое представление интенсивности отказов ($\lambda_э$) ЭРИ изделия представлено на рисунке 36 по данным таблицы 4. Данный рисунок позволяет наглядно отразить надежность всех элементов схемы. Из него видно, что самыми ненадежными элементами являются разъемные соединения. Это полностью согласовывается с теорией [79].

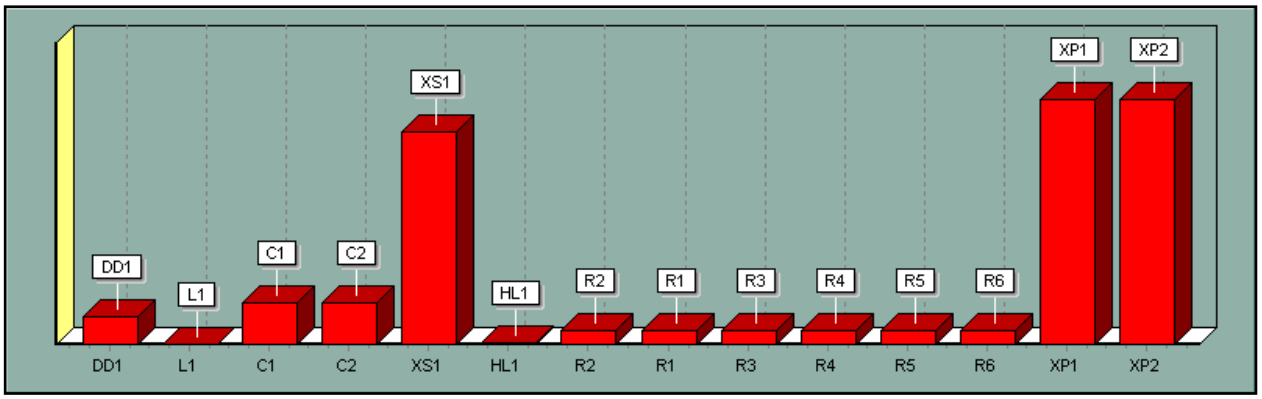


Рисунок 36 – Графическое представление интенсивности отказов ЭРИ изделия

В таблице 5 приведена эксплуатационная интенсивность отказов ЭРИ изделия от температуры окружающей среды.

Таблица 5 – Эксплуатационная интенсивность отказов ЭРИ изделия при различных температурах окружающей среды

№ п/п	Температура [°C]	Эксплуатационная интенсивность отказов ЭРИ изделия $\lambda_э$ [1/час]
1	25	$6{,}7729121155935 \cdot 10^{-7}$
2	30	$7{,}33408683117887 \cdot 10^{-7}$
3	35	$7{,}93278502442263 \cdot 10^{-7}$
4	40	$8{,}57087286476599 \cdot 10^{-7}$
5	45	$9{,}25035010368447 \cdot 10^{-7}$
6	50	$9{,}97335860120074 \cdot 10^{-7}$
7	55	$1{,}07429013951332 \cdot 10^{-6}$
8	60	$1{,}1560154007384 \cdot 10^{-6}$
9	65	$1{,}24294740544428 \cdot 10^{-6}$
10	70	$1{,}33532148538507 \cdot 10^{-6}$
11	75	$1{,}43355464388151 \cdot 10^{-6}$
12	80	$1{,}53809514813474 \cdot 10^{-6}$

На рисунке 36 приведено графическое представление эксплуатационной интенсивности отказов ЭРИ изделия от температуры окружающей среды.

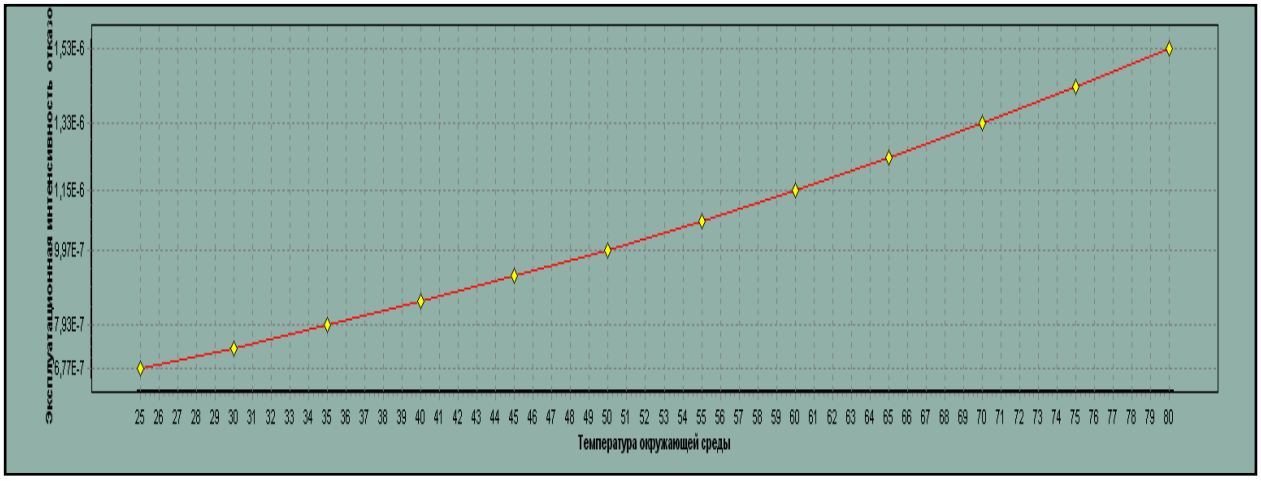


Рисунок 37 – Графическое представление эксплуатационной интенсивности отказов ЭРИ изделия от температуры окружающей среды

В таблице 6 приведена интенсивность отказов ЭРИ изделия в режиме хранения от температуры окружающей среды.

Таблица 6 – Интенсивность отказов ЭРИ изделия в режиме хранения при различных температурах окружающей среды

№ п/п	Температура [°C]	Интенсивность отказов ЭРИ изделия врежиме хранения λ, [1/час]
1	25	$5{,}28217203310533 \cdot 10^{-9}$
2	30	$5{,}6825473215838 \cdot 10^{-9}$
3	35	$6{,}1196954170839 \cdot 10^{-9}$
4	40	$6{,}59724141982565 \cdot 10^{-9}$
5	45	$7{,}11918479723295 \cdot 10^{-9}$
6	50	$7{,}68992247657252 \cdot 10^{-9}$
7	55	$8{,}38929548052647 \cdot 10^{-9}$
8	60	$9{,}08259778498177 \cdot 10^{-9}$
9	65	$9{,}83061099191218 \cdot 10^{-9}$
10	70	$1{,}06496287344271 \cdot 10^{-8}$
11	75	$1{,}15464793282172 \cdot 10^{-8}$
12	80	$1{,}25285488650206 \cdot 10^{-8}$

На рисунке 37 приведено графическое представление интенсивности отказов ЭРИ изделия в режиме хранения от температуры окружающей среды.

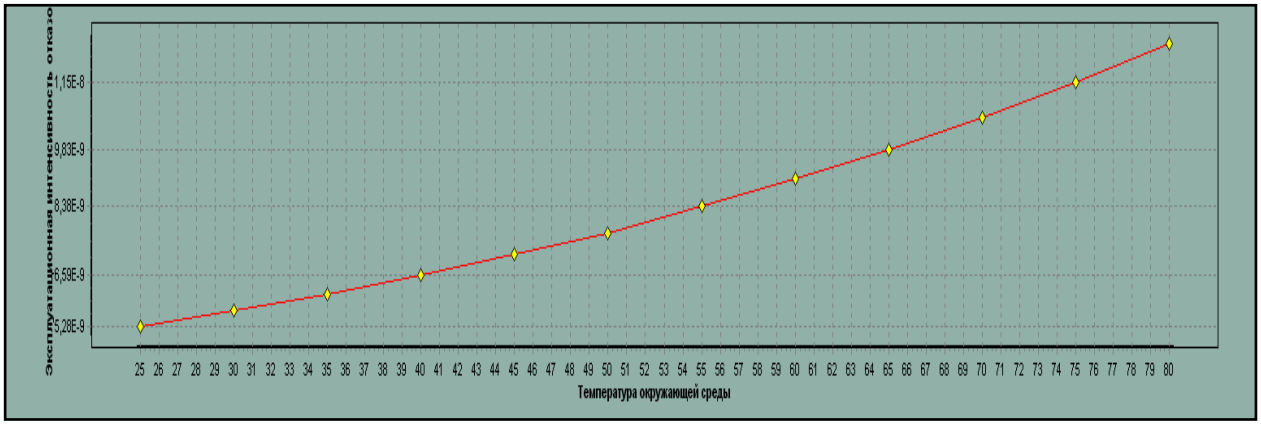


Рисунок 38 – Графическое представление интенсивности отказов ЭРИ изделия в режиме хранения от температуры окружающей среды

По результатам расчета:

- Срок хранения изделия – 189316016 час.
- Вероятность безотказной работы за 480 часов - 0,999674953114984 1/час.
- Среднее время наработки на отказ – 1476470 час.

По результатам расчета видно, что изделие надежно.

4.1.1.5. Расчет продолжительности работы от батареи.

Устройство ЭТИП является изделием, работающим автономно от батареи или аккумулятора. В связи с этим, время работы без подзарядки является важнейшим параметром изделия.

Современные электронные устройства − это сложные технические изделия, точный теоретический расчет энергопотребления которых чрезвычайно затруднен. Связано это прежде всего с тем, что энергопотребление компонентов, работающих под управлением микропрограмм (например, МК, радиомодули и д.р.) сильно зависит от структуры этих микропрограмм. Микропрограмма, выполняющая одну и ту же задачу, но написанная разными программистами (имеющими различную подготовку, опыт работы, использующих различные языки программирования, применяющие различные технологии и подходы при написании программы) может приводить к тому, что потребление МК, работающего под ее управлением, будет отличаться в разы.

Энергопотребление устройства ЭТИП измерялось нами по схеме включения, приведенной на рисунке 39.

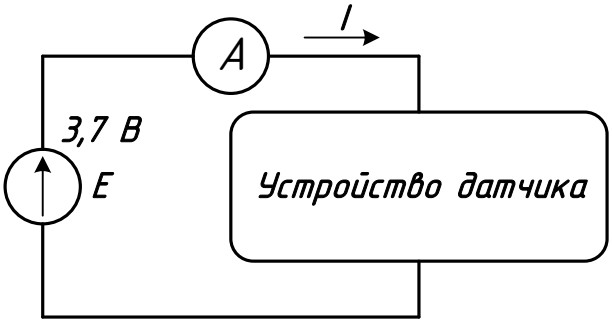


Рисунок 39 – Схема включения устройства ЭТИП для измерения его энергопотребления

Среднее энергопотребление с текущей версией ПО составило 3,5 мА ± 5%. Время непрерывной работы рассчитывалось по выражению:

$$N_{\text{Дней}} = \frac{C_{\text{АБ}}}{24 \cdot I}, \qquad (53)$$

где:

$N_{Дней}$ – количество дней непрерывной работы от аккумуляторной батареи;

$C_{АБ}$ – емкость аккумуляторной батареи в мА/ч, величина, постоянная для конкретного типа аккумуляторной батареи;

I – энергопотребление изделия, величина, постоянная для текущей версии ПО, равная 3,5 мА;

В устройстве ЭТИП используется аккумуляторная батарея типа 18650 (6000 мА/ч). Для данного типа батареи расчетное количество дней непрерывной работы ($N_{Дней}$) составляет 71,43 дня. На рисунке 40 представлена гистограмма с расчетным количеством дней непрерывной работы для аккумуляторных батарей различной емкости стандартного ряда.

Рисунок 40 – Зависимость количества дней непрерывной работы от емкости аккумуляторных батарей из стандартного ряда

Данная диаграмма применяется при выборе емкости аккумуляторной батареи для настройки устройства на детектирование заданного физиологического явления. Например, для надежного детектирования родов коровы устройство ЭТИП должно быть размещено не менее чем за семь дней до предполагаемого отела.

В перспективе, время автономной работы устройства ЭТИП может быть увеличено за счет использования менее энергопотребляющей элементной базы

(например, может быть использован микропотребляющий МК фирмы STMicroelectronics серии STM32L), более емких аккумуляторов и более эффективных программных алгоритмов работы устройства.

4.1.2. Устройство сопряжения с ПЭВМ.

4.1.2.1. Схема электрическая структурная.

Схема электрическая структурная устройства (рисунок 41) содержит следующие функциональные модули:

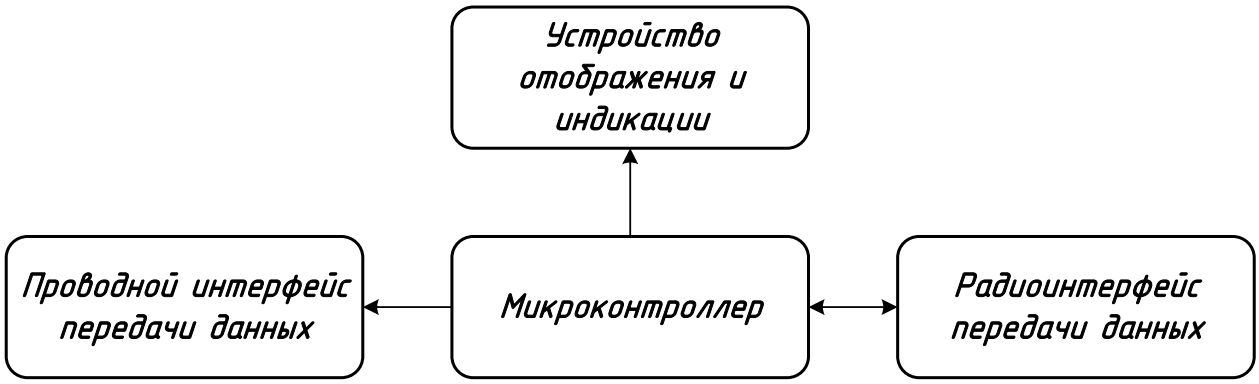


Рисунок 41 – Устройство сопряжения. Схема электрическая структурная

- Микроконтроллер с ядром AVR;
- Проводной интерфейс передачи данных;
- Радиоинтерфейс передачи данных;
- Устройство отображения и индикации;

Микроконтроллер является основным управляющим блоком схемы. Он управляет логикой получения данных от устройства ЭТИП, преобразованием полученных данных и передачу их в ПЭВМ. МК инициализирует радио- и проводной интерфейс передачи данных. Осуществляет преобразование информационных пакетов между ними.

Радиоинтерфейс передачи данных осуществляет прием полученной по радиоканалу информации на частоте 2,4 ГГц в нелицензируемом диапазоне частот ISM и передачу этой информации МК для дальнейшей обработки.

Устройство отображения и индикации является интерфейсом "человек-машина" и осуществляет отображение внутреннего состояния изделия способом, пригодным для непосредственного восприятия человеком.

Устройство ЭТИП обеспечивает:

- Прием пакетов данных от устройства ЭТИП;
- Преобразование пакетов данных, полученных по радиоинтерфейсу, в пакеты данных, пригодных для передачи по проводному интерфейсу в ПЭВМ;
- Индикация внутреннего состояния изделия во время работы и во внештатных ситуациях.

4.1.2.2. Обоснование выбора элементной базы.

Схема электрическая принципиальная (рисунок 44) построена с использованием следующих СБИС:

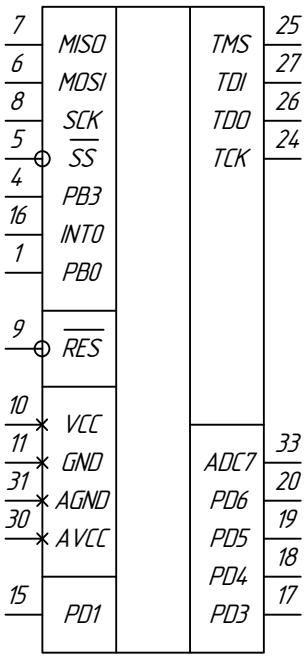


Рисунок 42 – Условно-графическое обозначение МК ATmega16-16MU

Отличительные особенности:

- 8-разрядное ядро;
- 32x8 рабочих РОН;

- Энергонезависимые памяти программ (износостойкость: 10 тысяч циклов стирания/записи) и данных;
- 16 Кбайт внутрисхемно-программируемой флэш-памяти программ;
- 1 Кбайт статического ОЗУ памяти данных;
- SPI интерфейс;
- Внутренний осциллятор до 8 МГц;
- Рабочие напряжения: 4,5 – 5,5 В;
- Рабочая температура: -40˚С...+85˚С.

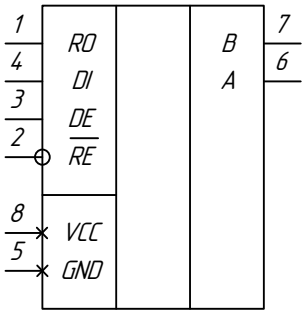


Рисунок 43 – Условно-графическое обозначение микросхемы интерфейса RS-485 ADM485EARZ

Отличительные особенности:

- Соответствие стандартам: TIA/EIA RS-485-/RS-422;
- Режим передачи данных: Полудуплекс;
- Скорость передачи данных: до 2,5 МБит/с;
- Рабочие напряжения: 5В ± 5%;
- Рабочая температура: -40˚С...+85˚С.

Все узлы изделия выполнены с применением стандартных СБИС, что обеспечивает малые габариты, энергопотребление, простоту и низкую стоимость изготовления, высокую надежность и, как следствие, большой период наработки изделия на отказ. Возможность модификации микропрограммного обеспечения изделия.

Применение распространенной и доступной элементной базы обеспечивает простоту реализации изделия. Все элементы согласуются друг с другом без дополнительных устройств.

Вышеуказанные свойства элементной базы обусловили ее выбор для разработки и изготовления изделия.

4.1.2.3. Схема электрическая принципиальная.

Рассмотрим работу изделия по схеме электрической принципиальной (рисунок 44) в положительной логике (U_{0max} = 0,7 В, U_{1min} = 4,2 В).

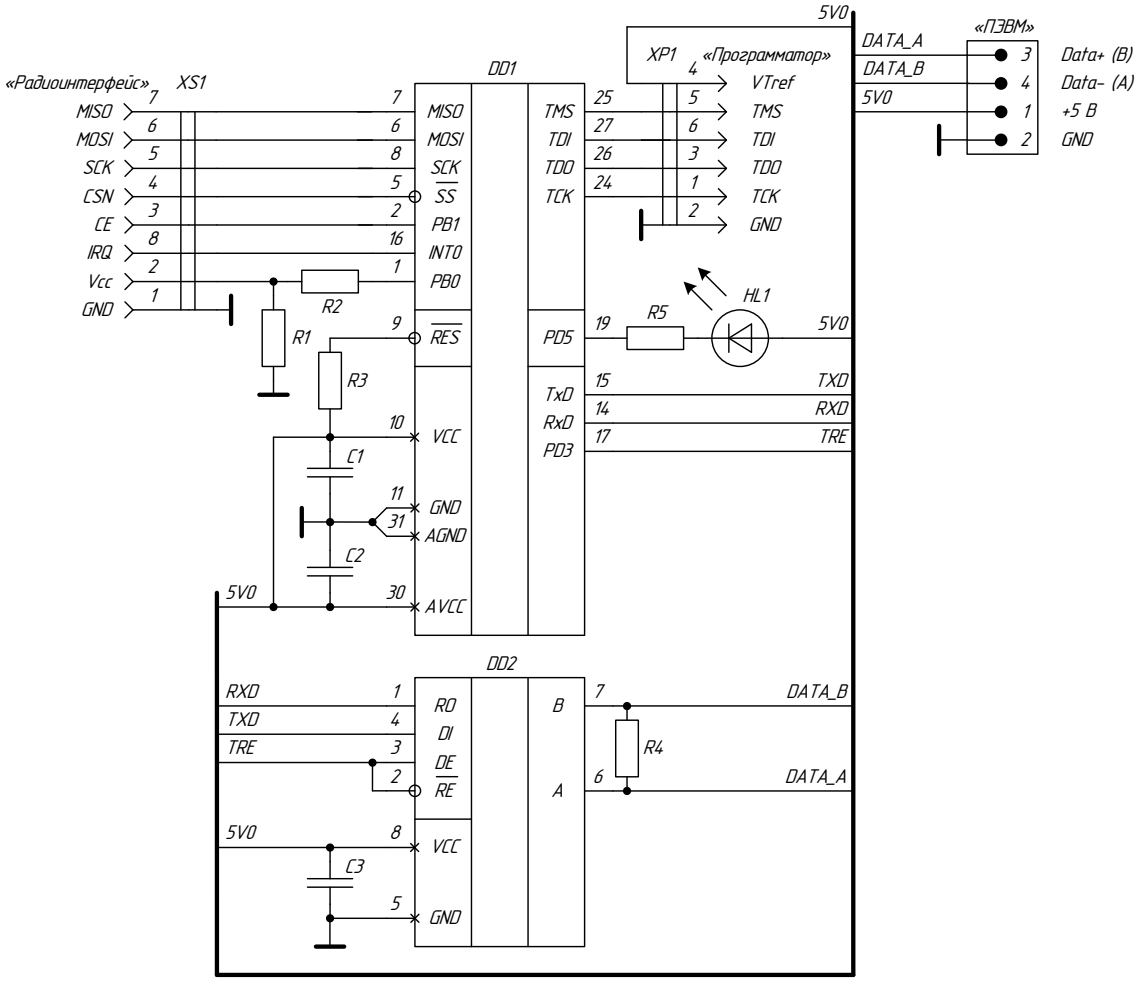


Рисунок 44 – Устройство сопряжения. Схема электрическая принципиальная

При подачи питания +5 В на контакт 1 контактной группы "ПЭВМ", с начальной задержкой в 65 мс, обусловленной стабилизацией внутреннего

тактового генератора, начинается выполнение заложенной в него микропрограммы.

Микропрограмма инициализирует модуль радиоинтерфейса, подключенный к разъему XS1 по интерфейсу SPI (выводы MISO, MOSI, SCK, CSN), и проводной интерфейс передачи данных RS-485 (DD2) (линии RXD, TXD, TRE).

После включения и инициализации радиоинтерфейса nRF24L01, и проводного интерфейса передачи данных RS-485 (DD2), МК (DD1) включает на 2 сек зеленый светодиод (HL1), индицируя о корректной работе изделия.

Далее с частотой следования данных, полученных по радиоинтерфейсу nRF24L01, МК (DD1) преобразует их в данные, пригодные для передачи по проводному интерфейсу передачи данных RS-485 (DD2), и передает их в ПЭВМ.

Мигание светодиода (HL1), отличное от приведенного выше, означает ошибку в микропрограмме МК (DD1).

Разъем XP1 предназначен для подключения программатора JTAG ICE MK2 к изделию и программирования МК (DD1) через промышленный интерфейс программирования и отладки JTAG (выводы TMS, TDI, TDO, TCK).

4.1.2.4. Расчет надежности.

Расчет надежности изделия выполнен с использованием программного комплекса АСОНИКА-К.

При расчете инженерной надежности изделия используются следующие исходные данные схемы электрической принципиальной (рисунок 44):

Таблица 7 – Интенсивность отказов ЭРИ изделия

№ п/п	Позиционное обозначение	Наименование ЭРИ	Интенсивность отказов эксплуатационная. $\lambda_э$ [1/час]	Интенсивность отказов в режиме ожидания. $\lambda_о$ [1/час]
1	C1 – C3	GCM155R71C104KA55D	$6{,}89\cdot10^{-10}$	$6{,}89\cdot10^{-12}$
2	DD1	ATMega16L-8MU	$1{,}08\cdot10^{-7}$	$3{,}05\cdot10^{-9}$
3	DD2	ADM485EARZ	$2{,}35\cdot10^{-8}$	$3{,}67\cdot10^{-10}$
4	R1 – R4	RG1005P	$1{,}05\cdot10^{-8}$	$1{,}05\cdot10^{-10}$
5	R5	RC0402JR-0722R	$1{,}05\cdot10^{-8}$	$1{,}05\cdot10^{-10}$
6	HL1	KPHHS-1005CGCK	$5{,}61\cdot10^{-10}$	$1{,}40\cdot10^{-11}$
7	XS1	PBD-20	$1{,}60\cdot10^{-7}$	$2{,}04\cdot10^{-9}$
8	XP1	M50-3600542	$1{,}84\cdot10^{-7}$	$4{,}61\cdot10^{-10}$

Общая интенсивность отказа ЭРИ равна $5{,}31067777840534 \cdot 10^{-7}$. Общая интенсивность отказа в режиме ожидания $6{,}47485208563917 \cdot 10^{-9}$.

Графическое представление интенсивности отказов ($\lambda_э$) ЭРИ изделия представлено на рисунке 45 по данным таблицы 7. Данный рисунок позволяет наглядно отразить надежность всех элементов схемы. Из него видно, что самыми ненадежными элементами являются разъемные соединения. Это полностью согласовывается с теорией [79].

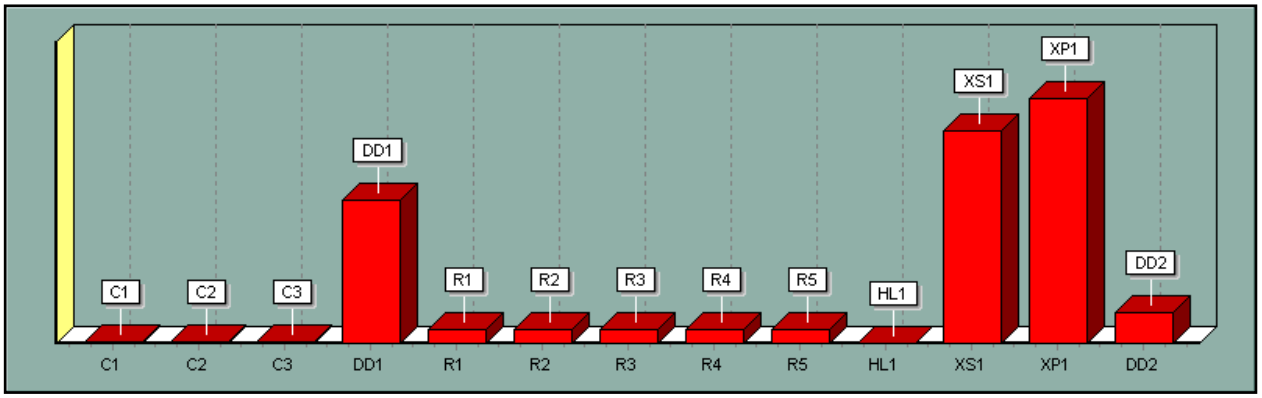


Рисунок 45 – Графическое представление интенсивности отказов ЭРИ изделия

В таблице 8 приведена эксплуатационная интенсивность отказов ЭРИ изделия от температуры окружающей среды.

Таблица 8 – Эксплуатационная интенсивность отказов ЭРИ изделия при различных температурах окружающей среды

№ п/п	Температура [°C]	Эксплуатационная интенсивность отказов ЭРИ изделия $\lambda_э$ [1/час]
1	25	$5{,}31067769315996 \cdot 10^{-7}$
2	30	$5{,}88958471325896 \cdot 10^{-7}$
3	35	$6{,}54522182230721 \cdot 10^{-7}$
4	40	$7{,}28865188648342 \cdot 10^{-7}$
5	45	$8{,}13240092156775 \cdot 10^{-7}$
6	50	$9{,}09058314846334 \cdot 10^{-7}$
7	55	$1{,}01790249118494 \cdot 10^{-6}$
8	60	$1{,}1415396556913 \cdot 10^{-6}$
9	65	$1{,}28193323689629 \cdot 10^{-6}$
10	70	$1{,}44125590395561 \cdot 10^{-6}$
11	75	$1{,}62190190167166 \cdot 10^{-6}$
12	80	$1{,}82649830549053 \cdot 10^{-6}$

На рисунке 45 приведено графическое представление эксплуатационной интенсивности отказов ЭРИ изделия от температуры окружающей среды.

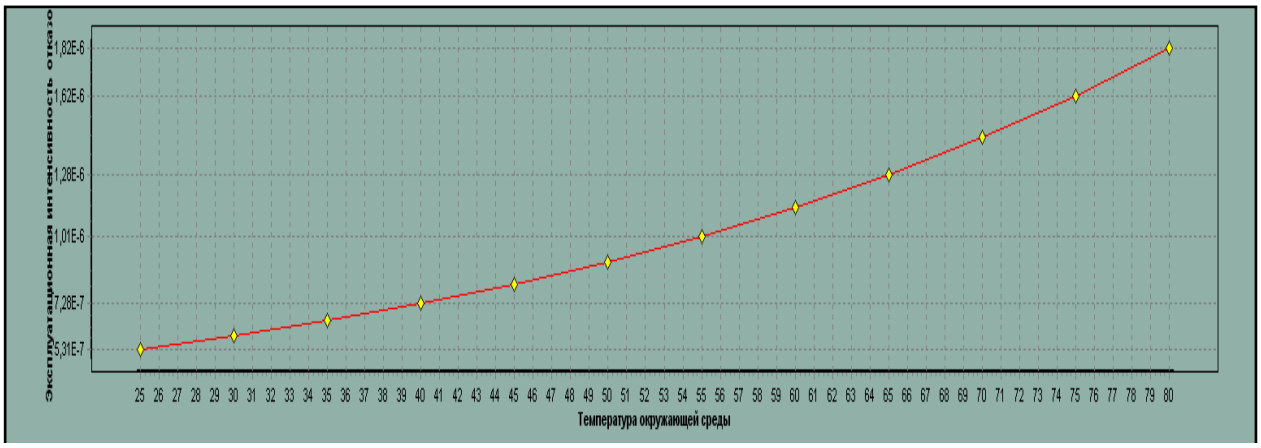


Рисунок 46 – Графическое представление эксплуатационной интенсивности отказов ЭРИ изделия от температуры окружающей среды

В таблице 9 приведена интенсивность отказов ЭРИ изделия в режиме хранения от температуры окружающей среды.

Таблица 9 – Интенсивность отказов ЭРИ изделия в режиме хранения при различных температурах окружающей среды

№ п/п	Температура [°C]	Интенсивность отказов ЭРИ изделия врежиме хранения $\lambda_э$ [1/час]
1	25	$6{,}47485265403702 \cdot 10^{-9}$
2	30	$7{,}24559034992467 \cdot 10^{-9}$
3	35	$8{,}15379941343508 \cdot 10^{-9}$
4	40	$9{,}22288112548131 \cdot 10^{-9}$
5	45	$1{,}04794430910715 \cdot 10^{-8}$
6	50	$1{,}1953572354173 \cdot 10^{-8}$
7	55	$1{,}36791111771117 \cdot 10^{-8}$
8	60	$1{,}56939403694878 \cdot 10^{-8}$
9	65	$1{,}80402501825938 \cdot 10^{-8}$
10	70	$2{,}07648120920112 \cdot 10^{-8}$
11	75	$2{,}39192505802066 \cdot 10^{-8}$
12	80	$2{,}75602811683484 \cdot 10^{-8}$

На рисунке 46 приведено графическое представление интенсивности отказов ЭРИ изделия в режиме хранения от температуры окружающей среды.

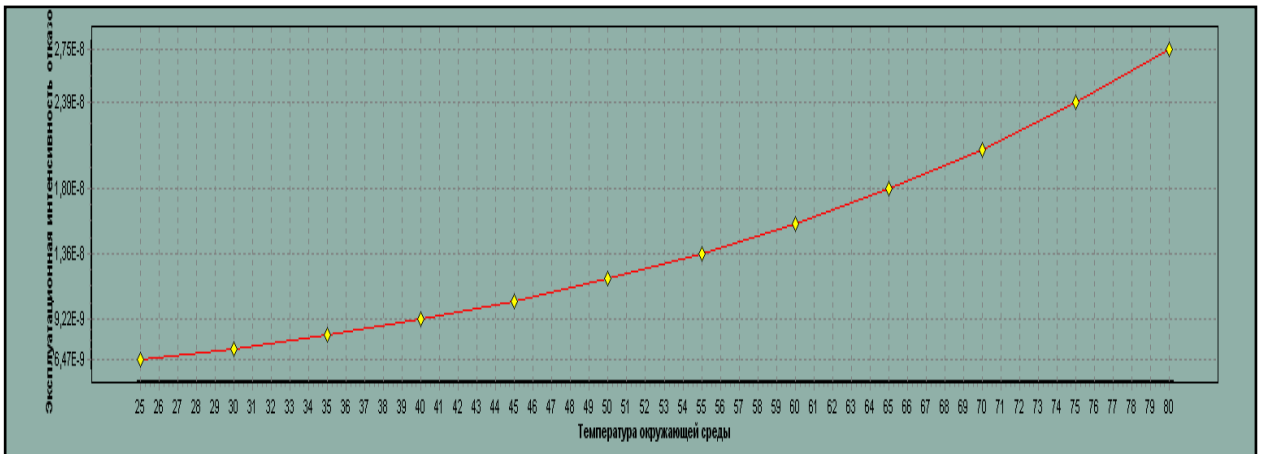


Рисунок 47 – Графическое представление интенсивности отказов ЭРИ изделия в режиме хранения от температуры окружающей среды

По результатам расчета:

- Срок хранения изделия – 154443682,5 час.
- Вероятность безотказной работы за 480 часов - 0,999745119954076 1/час.
- Среднее время наработки на отказ – 1882998,8 час.

По результатам расчета, видно, что изделие надежно.

4.2. Программная часть.

4.2.1. Устройства ЭТИП.

4.2.1.1. Обоснование выбора средств и методов программирования управляющего микроконтроллера.

Программы для МК производства Atmel с ядром AVR могут быть написаны на ЯНУ – «Assembler AVR», и ЯВУ – «C/C++».

Микропрограмма для МК (DD1) устройства ЭТИП была написана на ЯВУ "C". Это позволило существенно увеличить скорость разработки микропрограммы, снизить количество ошибок при написании исходного кода, ускорить процесс отладки, добиться гибкости применения отдельных программных модулей и переносимости программы на другие типы МК.

Применение ЯВУ при написании микропрограммы задействовало 19,8 % внутрисхемно-программируемой флэш-памяти программ и 4,2 % памяти данных статического ОЗУ.

Небольшое количество используемой памяти программ и данных МК позволит в будущем модернизировать существующую версию микропрограммы без изменения аппаратной части для введения новых режимов работы изделия (например, снижение энергопотребления), что сделает использование изделия более эффективным.

При написании микропрограммы использовались современные средства разработки и отладки (JTAG-эмулятор/отладчик, среда программирования AVR Studio 6), позволяющие просматривать текущее состояние регистров ядра, портов ввода/вывода и устройств МК в режиме пошаговой отладки. Благодаря этим средствам, отладка микропрограммы производилась непосредственно в работающем изделии, что существенно сократило количество ошибок программирования и время разработки.

Микропрограмма разрабатывалась по модульному принципу. Каждый модуль, отвечающий за определенную функцию, может быть перенесен на другой МК с ядром AVR практически без изменений (например, ATmega16, ATmega32, ATmega1286).

4.2.1.2. Описание логической структуры микропрограммы управляющего микроконтроллера.

Логическая структура микропрограммы управляющего МК представлена на рисунке 48.

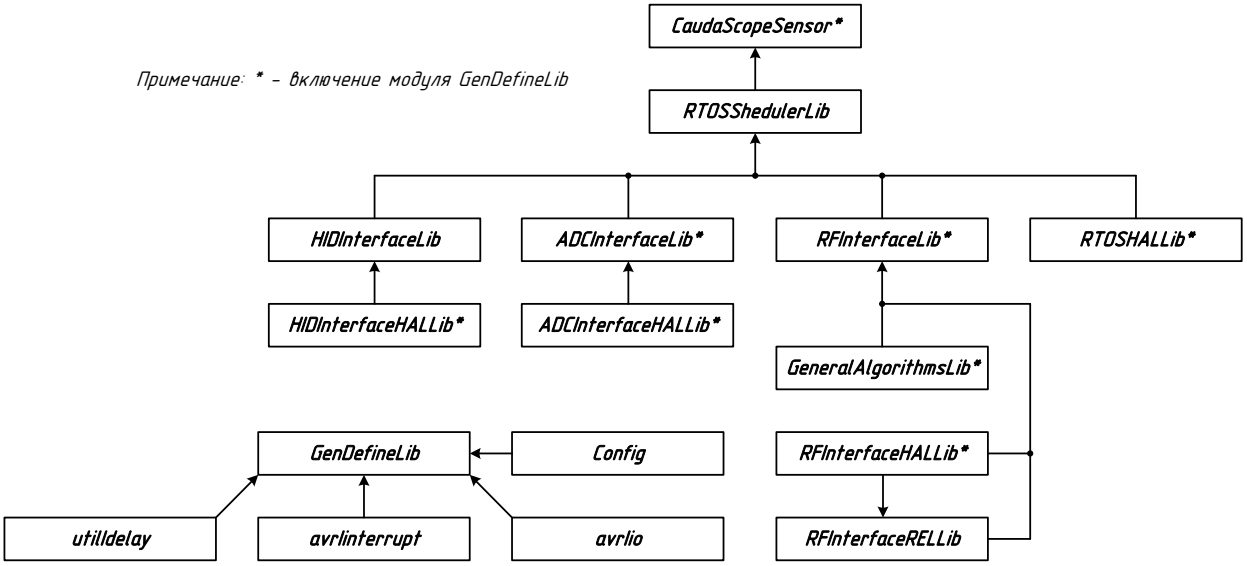


Рисунок 48 – Логическая структура микропрограммы устройства ЭТИП

Микропрограмма изделия имеет модульную структуру и состоит из следующих модулей:

- ADCInterfaceHALLib − Модуль абстракции АЦП датчика;
- ADCInterfaceLib − Модуль АЦП датчика;
- GeneralAlgorithmsLib − Модуль основных алгоритмов микропрограммы;
- HIDInterfaceLib − Модуль интерфейса взаимодействия с пользователем;
- CaudaScopeSensor − Основной файл микропрограммы;
- RFInterfaceHALLib − Файл модуля абстракции радиоинтерфейса от аппаратуры;
- RFInterfaceLib − Модуль радиоинтерфейса;
- RFInterfaceRELLib − Модуль доступа к регистрам радиоинтерфейса;
- RTOSHALLib − Модуль абстракции ОСРВ от аппаратуры;
- RTOSShedulerLib − Модуль планировщика задач ОСРВ.

После подачи электропитания на МК, микропрограммой инициализируются внутренние переменные модулей, запрещаются глобальные прерывания. Далее следует настройка периферии: АЦП, ЦАП, радиоинтерфейс передачи данных. Завершив настройку периферии, микропрограмма переходит бесконечный цикл, в котором осуществляется обработка прерываний от периферии.

Функции, выполняемые микропрограммой:

- Оцифровка сигнала с ТЧЭ;
- Удержание сигнала с ТЧЭ в рабочем диапазоне АЦП МК;
- Передача пакета данных с оцифрованными значениями сигнала с ТЧЭ устройству сопряжения с ПЭВМ по радиоинтерфейсу передачи данных;
- Индикация внутреннего состояния изделия во время работы и во внештатных ситуациях.

4.2.1.3. Описание функций программных модулей управляющего микроконтроллера.

Модуль ADCInterfaceHALLib

Программная реализация модуля находится в файлах ADCInterfaceHALLib.h, ADCInterfaceHALLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- uint16_t ADCIHHandler(void); − Функция обработки прерывания АЦП;
- void ADCIHInit(void); − Функция инициализации АЦП.

Модуль ADCInterfaceLib

Программная реализация модуля находится в файлах ADCInterfaceLib.h, ADCInterfaceLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- uint16_t ADCHandler(void); − Функция обработки прерывания АЦП;
- void ADCInitialization(void); − Функция инициализации АЦП;
- void ADCSampleStart(const enum ADCReistor_TypeDef); − Функция запуска выборки АЦП;
- void ADCSetResistorMatrixValue(const enum ADCReistor_TypeDef); − Функция установки общего сопротивления матрицы резисторов.

Модуль GeneralAlgorithmsLib

Программная реализация модуля находится в файлах GeneralAlgorithmsLib.h, GeneralAlgorithmsLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- float GAAverSumFunc(struct GAAverageBuffer_TypeDef*,const float); − Функция вычисляет среднее значение суммы чисел поступающих на ее вход методом скользящего окна;
- uint16_t ByteToUint16_t(const uint8_t*); − Функция получения числа типа "uint16_t" из массива байт;
- uint8_t GetPreambule(void); − Функция получения байта преамбулы;
- uint8_t GetXor(const uint8_t*,const uint8_t*); − Функция получения байта контрольной суммы массива байт;
- void Uint16_tToByte(uint8_t*,const uint16_t); − Функция получения массива байт из числа типа "uint16_t".

Модуль HIDInterfaceLib

Программная реализация модуля находится в файлах HIDInterfaceLib.h, HIDInterfaceLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- enum GeneralState_TypeDef HIDILEDGetState(void); − Функция получения состояния светодиода;
- void HIDIInitialization(void): − Функция инициализации интерфейса с пользователем;
- void HIDILEDChangeState(void); − Функция изменения состояния светодиода на противоположенное;
- void HIDILEDSetState(enum GeneralState_TypeDef); − Функция задания режима работы светодиода "Включено"/"Выключено".

Модуль CaudaScopeSensor

Программная реализация модуля находится в файле CaudaScopeSensor.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются функция int main(void);, являющаяся главной функцией микропрограммы.

Модуль RFInterfaceHALLib

Программная реализация модуля находится в файлах RFInterfaceHALLib.h, RFInterfaceHALLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- void RFIH_Hardware_Init(void); − Функция настройки аппаратного обеспечения для взаимодействия с радиомодулем;
- void RFIH_IRQ_Init(void); − Функция настройки запроса прерывания от радиомодуля;
- void RFIH_Port_Init(void); − Функция настройки портов для взаимодействия с радиомодулем;
- void RFIH_Powered(void); − Функция включения подачи питания на радиомодуль;
- void RFIH_SPI_Init(void); − Функция настройки интерфейса SPI;
- void RFIH_Unpowered(void); − Функция выключения подачи питания на радиомодуль;
- RFInterfaceLib
- int RFI_IRQ_Handler(void); − Функция обработки прерывания от радиомодуля;
- void RFI_Initialization(void); − Функция настройки радиомодуля;
- void RFI_Read_Device_Registers(void); − Функция опроса регистров радиомодуля;
- void RFI_Send_Buffer(const uint8_t*,const uint8_t); − Генерация передачи данных по радиоканалу.

Модуль RFInterfaceRELLib

Программная реализация модуля находится в файлах RFInterfaceRELLib.h, RFInterfaceRELLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- uint8_t RFIR_Get_Status(void); − Функция считывания регистра состояния радиомодуля;
- uint8_t RFIR_Read_Register(const uint8_t); − Функция чтения из регистра;
- uint8_t RFIR_WR_Byte(const uint8_t); − Функция передачи/получения байта данных в/от радиомодуля;
- void RFIR_Flush_Buffer(const enum RFIR_Command_TypeDef); − Функция очистки буферов приема и передачи радиомодуля;
- void RFIR_Read_Buffer(const uint8_t,uint8_t*,const uint8_t); − Функция чтения массива данных из радиомодуля;
- void RFIR_Run_Write(const uint8_t); − Функция запуска передачи данных;
- void RFIR_Write_Buffer(const uint8_t,const uint8_t*,const uint8_t); − Функция записи массива данных в радиомодуль;
- void RFIR_Write_Register(const uint8_t,const uint8_t); − Функция записи в регистр.

Модуль RTOSHALLib

Программная реализация модуля находится в файлах RTOSHALLib.h, RTOSHALLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующая функция:

- void RTOSHALTaskTimerInitialization(const uint16_t,const uint16_t); − Функция настройки системного таймера задач для МК.

Модуль RTOSShedulerLib

Программная реализация модуля находится в файлах RTOSShedulerLib.h, RTOSShedulerLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- void RTOSHardwareInitialization(void); − Функция настройки аппаратуры;
- void RTOSInitialization(void); − Функция настройки временного кванта планировщика;
- void RTOSIRQHandler(enum RTOSInterrupt_TypeDef); − Функция вызова обработчиков прерываний от аппаратуры;
- void RTOSTaskCounter(void); − Функция инкремента счетчика задач;
- void RTOSTaskFunction0(void); − Функция выполнения задачи 0;
- void RTOSTaskFunction1(void); − Функция выполнения задачи 1;
- void RTOSTaskHandler(void); − Функция запуска обработчиков задач;
- void SendDataToHost(const uint16_t, const enum ADCReistor_TypeDef); − Функция генерация пакета данных и его отправки на хост.

4.2.1.4. Протокол обмена информацией с устройством сопряжения с ПЭВМ.

Обмен данными ведется в симплексном режиме по радиоинтерфейсу передачи данных nRF24L01. Структура передаваемого байта приведена в таблице 10.

Таблица 10 – Структура байта, передаваемого с использованием радиоинтерфейса передачи данных nRF24L01

№ бита	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Байт	Старший значащий бит	-	-	-	-	-	-	Младший значащий бит

Изделие передает пакеты данных по готовности после включения питания с частотой (f) оцифровки сигнала с ТЧЭ (16 Гц). На рисунке 49 показана временная

диаграмма передачи пакета данных от устройства ЭТИП к устройству сопряжения с ПЭВМ.

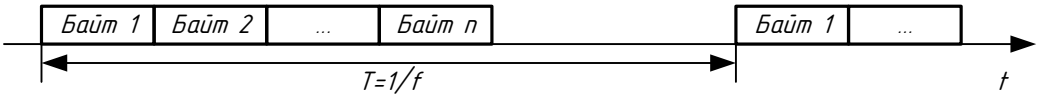


Рисунок 49 – Временная диаграмма передачи пакета данных от устройства ЭТИП к устройству сопряжения с ПЭВМ

В таблице 11 приведена структура пакета данных передаваемого от устройства ЭТИП к устройству сопряжения с ПЭВМ.

Таблица 11 – Структура пакета данных передаваемого от устройства ЭТИП к устройству сопряжения с ПЭВМ

№ поля по порядку следования в передаваемом пакете	Поле	Размер поля в байтах	Описание
1	STB	1	Маркер начала пакета (0xAA).
2	PREAMBULE	1	Используется для ускорения процесса синхронизации приемника и передатчика. Для четных пакетов значение поля равно 0x5A, для нечетных – 0xA5.
3	PAYLOAD	4	Полезная нагрузка пакета.
4	XOR	1	Контрольная сумма (XOR) поля «PAYLOAD».
5	SPB	1	Маркер окончания пакета (0x55).

Итого: 8 байт.

В таблице 12 приведена структура поля PAYLOAD для пакета данных, передаваемого от устройства ЭТИП к устройству сопряжения с ПЭВМ.

Таблица 12 – Структура поля PAYLOAD для пакета данных, передаваемого от устройства ЭТИП к устройству сопряжения с ПЭВМ

Поле	Тип поля	Байт	Описание
ADC_Data	Байтовое (uint16_t)	0 (Младший)	Оцифрованные данные с ТЧЭ.
		1 (Старший)	
Resist	Байтовое (uint8_t)	2	Общее значение сопротивления в модуле ЦАП.
Counter	Байтовое (uint8_t)	3	Счетчик пакетов данных. Применяется при мониторинге потерянных пакетов данных.

Итого: 4 байта.

4.2.2. Устройство сопряжения с ПЭВМ.

4.2.2.1. Обоснование выбора средств и методов программирования управляющего микроконтроллера.

Программы для МК производства Atmel с ядром AVR могут быть написаны на ЯНУ – «Assembler AVR», и ЯВУ – «C/C++».

Микропрограмма для МК (DD1) устройства ЭТИП была написана на ЯВУ "C". Это позволило существенно увеличить скорость разработки микропрограммы, снизить количество ошибок при написании исходного кода, ускорить процесс отладки, добиться гибкости применения отдельных программных модулей и переносимости программы на другие типы МК.

Применение ЯВУ при написании микропрограммы задействовало 18,2 % внутрисхемно-программируемой флэш-памяти программ и 5 % памяти данных статического ОЗУ.

Небольшое количество используемой памяти программ и данных МК позволит в будущем модернизировать существующую версию микропрограммы без изменения аппаратной части для введения новых режимов работы изделия, что сделает использование изделия более эффективным.

При написании микропрограммы использовались современные средства разработки и отладки (JTAG-эмулятор/отладчик, среда программирования AVR Studio 6), позволяющие просматривать текущее состояние регистров ядра, портов ввода/вывода и устройств МК в режиме пошаговой отладки. Благодаря этим средствам, отладка микропрограммы производилась непосредственно в работающем изделии, что существенно сократило количество ошибок программирования и время разработки.

Микропрограмма разрабатывалась по модульному принципу. Каждый модуль, отвечающий за определенную функцию, может быть перенесен на другой МК с ядром AVR практически без изменений (например, ATmega16, ATmega32, ATmega1286).

4.2.3. Описание логической структуры микропрограммы управляющего микроконтроллера.

Логическая структура микропрограммы управляющего МК представлена на рисунке 50.

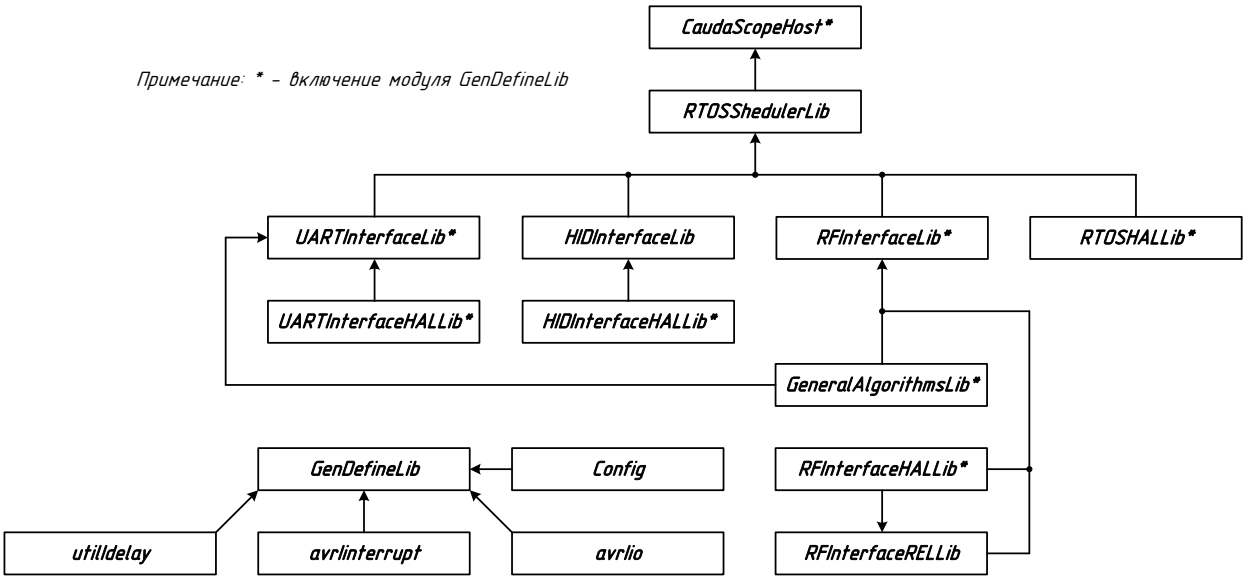


Рисунок 50 – Логическая структура микропрограммы устройства сопряжения с ПЭВМ

Микропрограмма изделия имеет модульную структуру и состоит из следующих модулей:

- GeneralAlgorithmsLib − Модуль основных алгоритмов микропрограммы;
- HIDInterfaceLib − Модуль интерфейса взаимодействия с пользователем;
- CaudaScopeHost − Основной файл микропрограммы;
- RFInterfaceHALLib − Модуль абстракции радиоинтерфейса от аппаратуры;
- RFInterfaceLib − Модуль радиоинтерфейса;
- RFInterfaceRELLib − Модуль доступа к регистрам радиоинтерфейса;
- RTOSHALLib − Модуль абстракции ОСРВ от аппаратуры;
- RTOSShedulerLib − Модуль планировщика задач ОСРВ;
- UARTInterfaceHALLib − Модуль абстракции UART от аппаратуры;
- UARTInterfaceLib − Модуль UART.

После подачи электропитания на МК микропрограммой инициализируются внутренние переменные модулей, запрещаются глобальные прерывания. Далее

следует настройка периферии: проводной интерфейс передачи данных, радиоинтерфейс передачи данных. Завершив настройку периферии, микропрограмма переходит бесконечный цикл, в котором осуществляется обработка прерываний от периферии.

Функции, выполняемые микропрограммой:

- Прием пакета данных с оцифрованными значениями сигнала с ТЧЭ от устройства ЭТИП по радиоинтерфейсу передачи данных;
- Преобразование полученного пакета данных в формат, пригодный для передачи в ПЭВМ;
- Передача пакета данных с оцифрованными значениями сигнала с ТЧЭ в ПЭВМ по проводному интерфейсу передачи данных;
- Индикация внутреннего состояния изделия во время работы и внештатных ситуациях.

4.2.3.1. Описание функций программных модулей управляющего микроконтроллера.

Модуль GeneralAlgorithmsLib

Программная реализация модуля находится в файлах GeneralAlgorithmsLib.h, GeneralAlgorithmsLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- float GAAverSumFunc(struct GAAverageBuffer_TypeDef*,const float); − Функция вычисляет среднее значение суммы чисел поступающих на ее вход методом скользящего окна;
- uint16_t ByteToUint16_t(const uint8_t*); − Функция получения числа типа "uint16_t" из массива байт;
- uint8_t GetXor(const uint8_t*,const uint8_t*); − Функция получения байта контрольной суммы массива байт;
- void Uint16_tToByte(uint8_t*,const uint16_t); − Функция получения массива байт из числа типа "uint16_t".

Модуль HIDInterfaceLib

Программная реализация модуля находится в файлах HIDInterfaceLib.h, HIDInterfaceLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- enum GeneralState_TypeDef HIDILEDGetState(void); − Функция получения состояния светодиода;
- void HIDIInitialization(void); − Функция инициализации интерфейса с пользователем;
- void HIDILEDChangeState(void); − Функция изменения состояния светодиода на противоположное;
- void HIDILEDSetState(enum GeneralState_TypeDef); − Функция задания режима работы светодиода "Включено"/"Выключено".

Модуль CaudaScopeHost

Программная реализация модуля находится в файле CaudaScopeHost.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются функция int main(void);, являющаяся главной функцией микропрограммы;

Модуль RFInterfaceHALLib

Программная реализация модуля находится в файлах RFInterfaceHALLib.h, RFInterfaceHALLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- void RFIH_Hardware_Init(void); − Функция настройки аппаратного обеспечения для взаимодействия с радиомодулем;
- void RFIH_IRQ_Init(void); − Функция настройки запроса прерывания от радиомодуля;
- void RFIH_Port_Init(void); − Функция настройки портов для взаимодействия с радиомодулем;

- void RFIH_Powered(void); − Функция включения подачи питания на радиомодуль;
- void RFIH_SPI_Init(void); − Функция настройки интерфейса SPI;
- void RFIH_Unpowered(void); − Функция выключения подачи питания на радиомодуль.

Модуль RFInterfaceLib

Программная реализация модуля находится в файлах RFInterfaceLib.h, RFInterfaceLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- int RFI_IRQ_Handler(void); − Функция обработки прерывания от радиомодуля;
- void RFI_Get_Recv_Data(uint8_t*,const uint8_t); − Функция передачи полученных данных;
- void RFI_Initialization(void); − Функция настройки радиомодуля;
- void RFI_Read_Device_Registers(void); − Функция опроса регистров радиомодуля.

Модуль RFInterfaceRELLib

Программная реализация модуля находится в файлах RFInterfaceRELLib.h, RFInterfaceRELLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- uint8_t RFIR_Get_Status(void); − Функция считывания регистра состояния радиомодуля;
- uint8_t RFIR_Read_Register(const uint8_t); − Функция чтения из регистра;
- uint8_t RFIR_WR_Byte(const uint8_t); − Функция передачи/получения байта данных в/от радиомодуля;
- void RFIR_Flush_Buffer(const enum RFIR_Command_TypeDef); − Функция очистки буферов приема и передачи радиомодуля;

- void RFIR_Read_Buffer(const uint8_t,uint8_t*,const uint8_t); − Функция чтения массива данных из радиомодуля;
- void RFIR_Run_Write(const uint8_t); − Функция запуска передачи данных;
- void RFIR_Write_Buffer(const uint8_t,const uint8_t*,const uint8_t); − Функция записи массива данных в радиомодуль;
- void RFIR_Write_Register(const uint8_t,const uint8_t); − Функция записи в регистр.

Модуль RTOSHALLib

Программная реализация модуля находится в файлах RTOSHALLib.h, RTOSHALLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующая функция: void RTOSHALTaskTimerInitialization(const uint16_t,const uint16_t); − функция настройки системного таймера задач для МК.

Модуль RTOSShedulerLib

Программная реализация модуля находится в файлах RTOSShedulerLib.h, RTOSShedulerLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- static void RTOSInitialization(void); − Функция настройки временного кванта планировщика;
- static void RTOSTaskFunction0(void); − Функция выполнения задачи 0;
- void RTOSHardwareInitialization(void); − Функция настройки аппаратуры;
- void RTOSIRQHandler(enum RTOSInterrupt_TypeDef); − Функция вызова обработчиков прерываний от аппаратуры;
- void RTOSTaskCounter(void); − Функция инкремента счетчика задач;
- void RTOSTaskHandler(void); − Функция запуска обработчиков задач.

Модуль UARTInterfaceHALLib

Программная реализация модуля находится в файлах UARTInterfaceHALLib.h, UARTInterfaceHALLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующая функция: void UIH_UART_Init(void); − функция настраивает асинхронный приемопередатчик.

Модуль UARTInterfaceLib

Программная реализация модуля находится в файлах UARTInterfaceLib.h, UARTInterfaceLib.c, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- int UI_UART_Send_Data(const uint16_t,const uint8_t, const uint8_t); − Функция передачи пакета информации на хост;
- void UI_UART_Initialization(void); − Функция настраивает УАПП;
- void UI_UART_IRQ_Handler(void); − Функция обработки прерывания от УАПП.

4.2.3.2. Протокол обмена информацией с ПЭВМ.

Обмен данными ведется в симплексном режиме по проводному интерфейсу передачи данных RS−485. Структура передаваемого байта приведена в таблице 13.

Таблица 13 – Структура байта, передаваемого с использованием проводного интерфейса передачи данных RS−485

№ бита	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Байт	Старший значащий бит	–	–	–	–	–	–	Младший значащий бит

Изделие передает пакеты данных по готовности после получения пакета данных от устройства ЭТИП. На рисунке 51 показана временная диаграмма передачи пакета данных от устройства сопряжения к ПЭВМ.

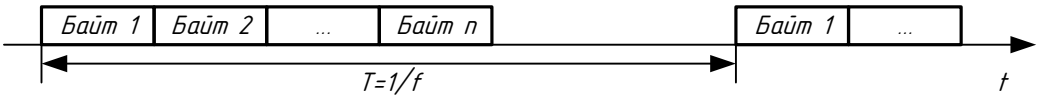


Рисунок 51 – Временная диаграмма передачи пакета данных от устройства сопряжения к ПЭВМ

В таблице 14 приведена структура пакета данных, передаваемого от устройства сопряжения к ПЭВМ.

Таблица 14 – Структура пакета данных, передаваемого от устройства сопряжения к ПЭВМ

№ поля по порядку следования в передаваемом пакете	**Поле**	**Размер поля в байтах**	**Описание**
1	STB	1	Маркер начала пакета (0xAA).
2	PREAMBULE	1	Используется для ускорения процесса синхронизации приемника и передатчика. Для четных пакетов значение поля равно 0x5A, для нечетных – 0xA5.
3	PAYLOAD	4	Полезная нагрузка пакета.
4	XOR	1	Контрольная сумма (XOR) поля «PAYLOAD».
5	SPB	1	Маркер окончания пакета (0x55).

Итого: 8 байт.

В таблице 15 приведена структура поля PAYLOAD для пакета данных, передаваемого от устройства сопряжения к ПЭВМ.

Таблица 15 – Структура поля PAYLOAD для пакета данных, передаваемого от устройства сопряжения к ПЭВМ

Поле	**Тип поля**	**Байт**	**Описание**
ADC_Data	Байтовое (uint16_t)	0 (Младший)	Оцифрованные данные с ТЧЭ.
		1 (Старший)	
Resist	Байтовое (uint8_t)	2	Общее значение сопротивления в модуле ЦАП.
Counter	Байтовое (uint8_t)	3	Счетчик пакетов данных. Применяется при мониторинге потерянных пакетов данных.

Итого: 4 байта.

4.2.4. Программа интерпретации данных для ПЭВМ.

4.2.4.1. Обоснование выбора средств и методов программирования.

СПО для ПЭВМ была написана на ЯВУ "C#". Это позволило существенно увеличить скорость разработки микропрограммы, снизить количество ошибок при написании исходного кода, ускорить процесс отладки, добиться гибкости применения отдельных программных модулей и переносимости программы на другие операционный системы (ОС) фирмы Microsoft.

Использование стандартного инструментария Microsoft позволяет получить переносимость разработанного программного обеспечения между различными версиями ОС Windows, что дает возможность пользователям использовать данное ПО с минимальными затратами усилий на его установку и настройку. Так же использование данного инструментария позволяет относительно просто модернизировать уже созданное ПО, расширяя базу идентифицируемых физиологических явлений и улучшать алгоритмы идентификации этих явлений.

СПО разрабатывалось по модульному принципу. При написании использовались современные средства разработки и отладки (среда программирования, разработки и отладки – Microsoft Visual Studio Express 2012, библиотеки Microsoft .NET Framework 4.5 и AForge 2.2.5), позволяющие просматривать текущее состояние переменных и массивов в режиме пошаговой отладки. Благодаря этим средствам, отладка СПО производилась непосредственно с рабочими данными, поступившими от устройства ЭТИП, что существенно сократило количество ошибок программирования и время разработки.

4.2.4.2. Описание логической структуры программы.

Логическая структура СПО ПЭВМ представлена на рисунке 52.

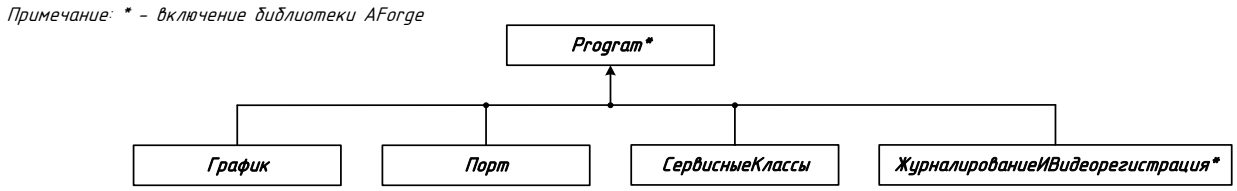


Рисунок 52 – Логическая структура СПО ПЭВМ

СПО ПЭВМ имеет модульную структуру и состоит из следующих модулей:

- Program.cs – Модуль основного файла программы;
- График.cs – Модуль файла графика;
- ЖурналированиеИВидеорегистрация.cs – Модуль файла журналирования и видеорегистрации;
- Порт.cs – Модуль файла последовательного порта;
- СервисныеКлассы.cs – Модуль файла сервисных классов и функций.

После включения СПО требуется включить соединение с устройством сопряжения. При удачном соединении СПО принимает пакеты с данными с частотой их передачи. Распаривает полученный пакет, проверяет его на наличие ошибок, при отсутствии ошибок передает дальше на следующий этап обработки, а при наличии ошибок – пропускает. Извлеченные данные поступают на основной модуль обработки, где обрабатываются в соответствии с алгоритмом идентификации явлений и отображаются на экране в виде графика. При идентификации явления, алгоритмом обработки генерируется сигнал «Идентификация». Это означает, что требуемое явление зафиксировано и информация о событии с идентификационным номером животного может быть передана в виде SMS−сообщения ветеринару или фермеру. Одновременно с этим включается запись видеоизображения (если это требуется и если подключена видеокамера), которое может быть отправлено ветеринару или фермеру в виде MMS−сообщения для более полного представления о идентифицированном событии и принятии адекватного решения. Скриншот СПО во время работы приведен на рисунке 53.

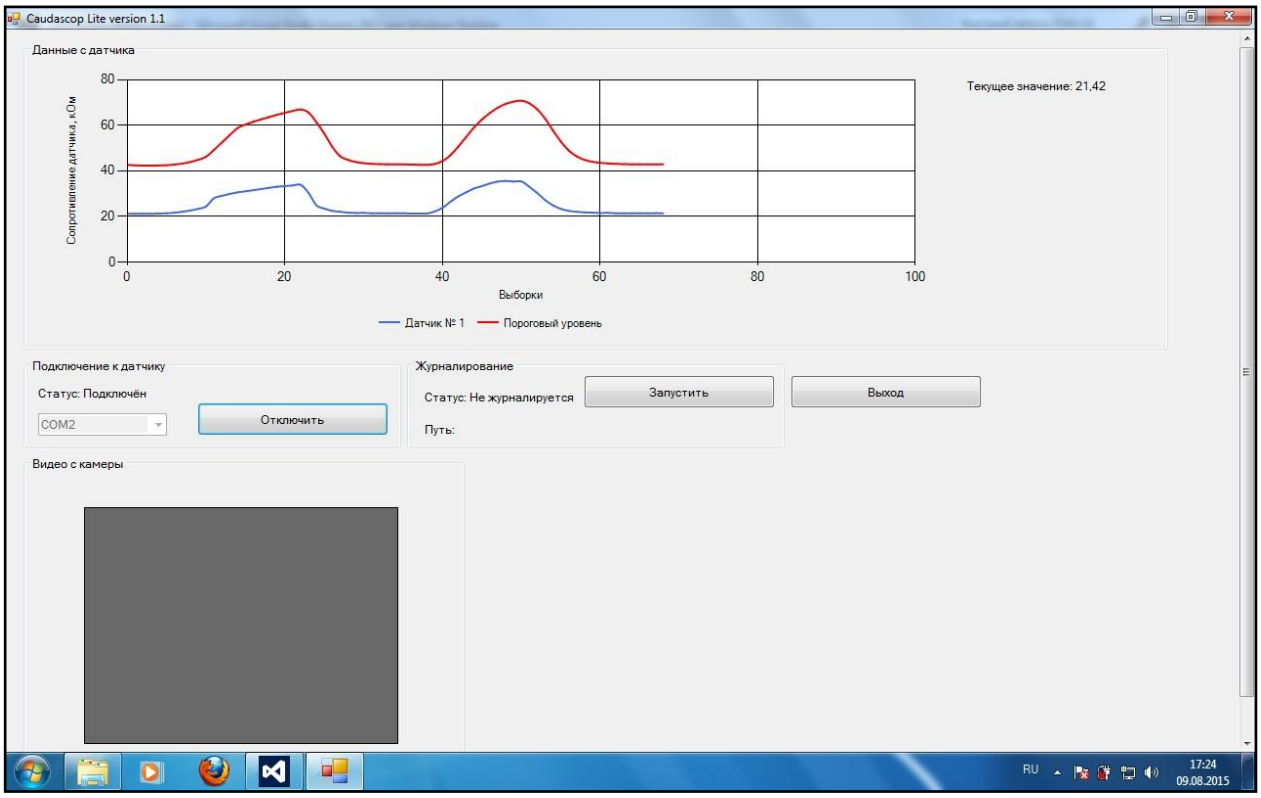


Рисунок 53 – СПО ПЭВМ во время работы

Функции, выполняемые СПО:

- Прием пакета данных с оцифрованными значениями сигнала с ТЧЭ от устройства сопряжения по проводному интерфейсу передачи данных;
- Идентификация пакетов, содержащих ошибки;
- Идентификация явления, происходящего с хвостом животного;
- Оповещение об идентификации требуемого явления, происходящего с хвостом животного.

4.2.4.3. Описание функций программных модулей.

Модуль Program

Программная реализация модуля находится в файле Program.cs, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- internal void НовыйКадр(object, NewFrameEventArgs); − Функция−обработчик события получения нового кадра с видеокамеры;
- private void serialPort1_DataReceived(object, SerialDataReceivedEventArgs); − Функция−обработчик события получения байта по последовательному порту;
- private void ВидеорегистрацияТаймер_Tick(object, EventArgs); − Функция−обработчик события тика таймера;
- private void КнопкаВыход_Click(object, EventArgs); − Функция−обработчик события нажатия на кнопку "Выход";
- private void КнопкаЖурналирование_Click(object, EventArgs); − Функция−обработчик события нажатия на кнопку "Журналирование";
- private void КнопкаПодключения_Click(object, EventArgs); − Функция−обработчик события нажатия на кнопку "Подключение";
- private void ОсновнойПрограммныйМодуль(string,double); − Функция основного модуля идентификации явлений, происходящих с хвостом животного;
- private void ФормаОсновная_FormClosing(object, FormClosingEventArgs); − Функция−обработчик события закрытия основной формы;
- private void ФормаОсновная_Load(object, EventArgs); − Функция−обработчик события загрузки основной формы;
- public void ВыходИзПрограммы(); − Функция−обработчик события выхода из программы.

Модуль График

Программная реализация модуля находится в файле График.cs, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- static public bool ДобавитьТочку(ФормаОсновная, double); − Функция добавления точки на график;
- static public void Очистить(ФормаОсновная); − Функция очистки графика.

Модуль ЖурналированиеИВидеорегистрация

Программная реализация модуля находится в файле ЖурналированиеИВидеорегистрация.cs, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- static private string ИндексированныйПуть(Тип Журналирования); − Функция создания индексированного пути к файлу;
- static public bool Деинициализация(ФормаОсновная); − Функция деинициализации журналирования;
- static public bool Запуск(ФормаОсновная); − Функция запуска журналирования;
- static public bool Инициализация(ФормаОсновная); − Функция инициализации журналирования;
- static public bool Остановка(ФормаОсновная); − Функция остановки журналирования;
- static public void ДобавитьДанные(Форма Основная, string, double); − Функция добавления кадра в видеофайл;
- static public void ДобавитьДанныеПотока(Форма Основная, string, double); − Функция добавления новых данных с ТЧЭ в файл;
- static public void ДобавитьКадр(Форма Основная, Bitmap); − Функция добавления кадра в видеофайл.

Модуль Порт

Программная реализация модуля находится в файле Порт.cs, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- static private double ПарсерПоляСопротивление(byte); − Функция парсинга поля сопротиления;

- static private void ОчиститьБуферПринятыхДанных(ФормаОсновная); − Функция очистки буфера принятых данных последовательного порта;
- static public bool ОбработчикПринятыхБайт(ФормаОсновная); − Функция обработчика принятых байт по последовательному порту;
- static public void Включить(ФормаОсновная); − Функция выключения последовательного порта и обработчика принятых байт;
- static public void Выключить(ФормаОсновная); − Функция включения последовательного порта и обработчика принятых байт.

Модуль СервисныеКлассы

Программная реализация модуля находится в файле СервисныеКлассы.cs, листинг приведен в приложении. Программный код снабжен подробными комментариями.

В модуле используются следующие функции:

- public byte GetXOR(byte[], int); − Функция вычисления контрольной суммы массива байт;
- public double GetRandomvalue(int); − Функция генерации случайного числа;
- public double ДобавитьДанные(double); − Функция добавления данных в усредняющий фильтр;
- public string GetCurrentData(); − Функция получения текущей даты;
- public string GetCurrentDataTime(); − Функция получения текущей даты и времени;
- public string GetCurrentFormatDataTime(); − Функция получения формата текущей даты и времени;
- public string GetCurrentFormatDate(); − Функция получения формата текущей даты;
- public string GetCurrentFormatTime(); − Функция получения формата текущего времени;
- public string GetCurrentTime(); − Функция получения текущего времени;

- public UInt16 GetRandomvalue(); − Функция генерации случайного числа типа "UInt16";
- public UInt16 GetRandomvalue(UInt16, UInt16); − Функция генерации случайного числа типа "UInt16";
- public Усредняющий_Фильтр(int); − конструктор класса Усредняющий_Фильтр.

ГЛАВА 5. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ.

5.1. Результаты исследований ЭТС на модели хвоста животного.

Из проведенных экспериментальных исследований на модели хвоста животного следует, что чувствительность исследованного ТЧЭ составляет: $k_ч$ = 67 %/(н/м^2).

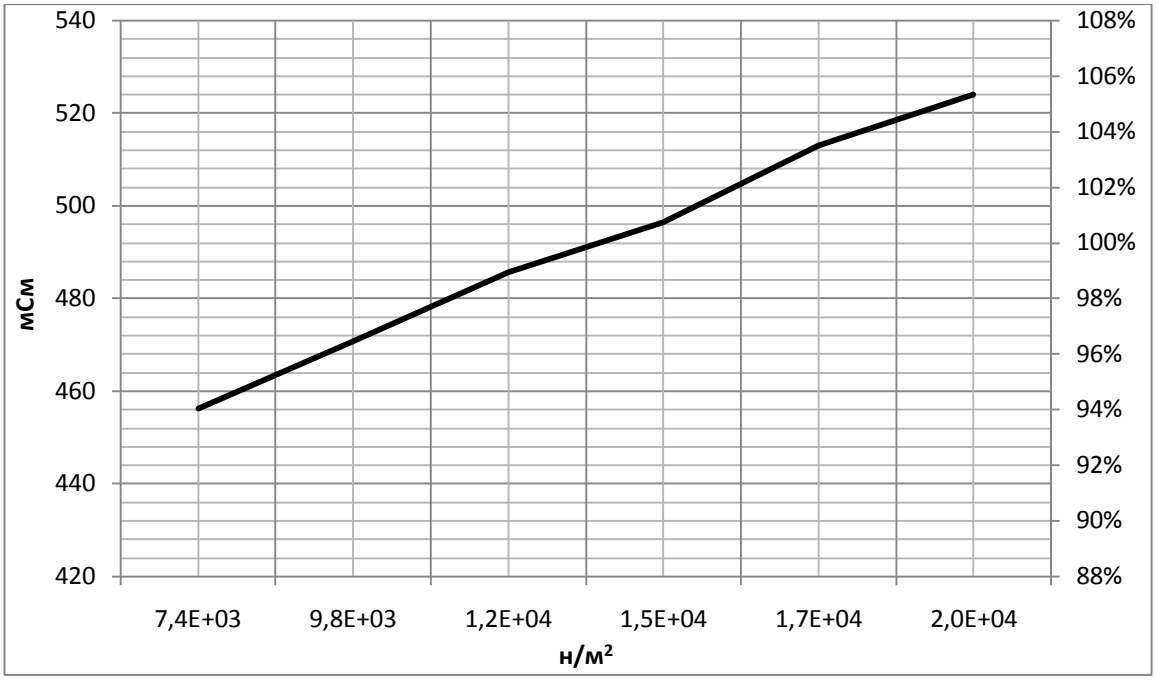


Рисунок 54 – Статические показания ТЧЭ (на выходе АЦП), нормированные на максимальное значение избыточного давления в камере ($2\cdot10^4$ н/м^2) при радиальном воздействии силы растяжения на модели хвоста животного

Эксперименты на модели хвоста животного позволили определить статическую характеристику ТЧЭ при радиальном воздействии силы растяжения (рисунок 54) и установить рабочий диапазон ТЧЭ.

5.2. Результаты исследований ЭТС при измерении мышечного напряжения на руке человека.

Реакция ТЧЭ на различные воздействия представлена на рисунках 55…57.

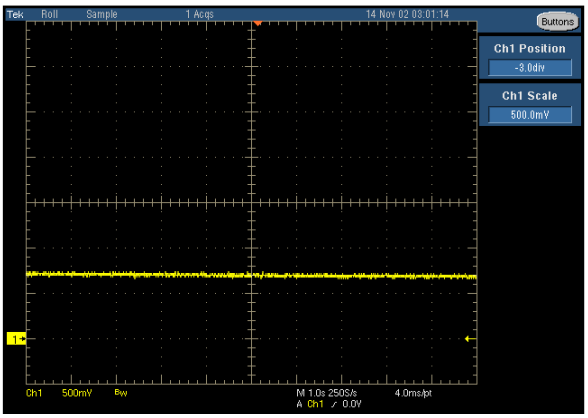


Рисунок 55 – Реакция ТЧЭ на отсутствие воздействия

На рисунке 55 представлена реакция ТЧЭ на отсутствие воздействия. Как видно из рисунка, сопротивление материала ($R_{ТЧЭ}(d)$) не изменяется (остается равным $R_{ТЧЭ}(d) = 7$ *кОм*, при $I_0 = 0{,}1$ *мА* и $d = const$), отсутствуют какие либо флуктуации и возмущения – характеристики материала ТЧЭ стабильны.

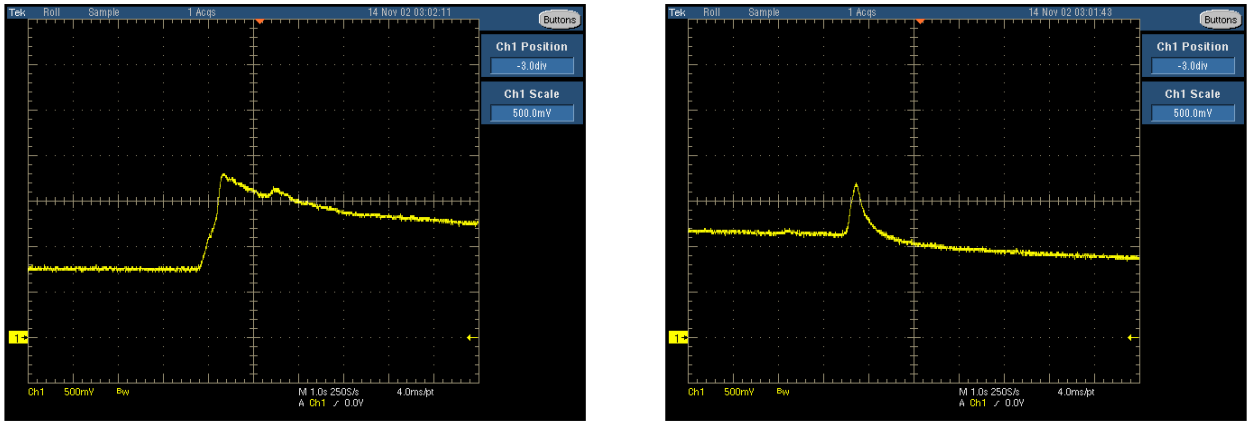


Рисунок 56 – Реакция ТЧЭ на воздействия. Слева – напряжение мышц предплечья, имитирующее напряжение мышц корня хвоста животного. Справа – расслабление мышц предплечья, имитирующее расслабление мышц корня хвоста животного

На рисунке 56 приведена реакция ТЧЭ на базовые воздействия: 1 – напряжение мышц; 2 – расслабление мышц. Как видно из рисунков, при напряжении мышц происходит резкое увеличение сопротивления материала ТЧЭ ($R_{ТЧЭ}(d) = 18$ *кОм*, при $I_0 = 0{,}1$ *мА*), превышающее почти в 2,5 раза сопротивление ТЧЭ при отсутствии воздействия.

Напряженное состояние мышц характеризуется сигналом, имеющим следующие параметры: $max(R_{ТЧЭ}(d)) = 14$ *кОм*, $min(R_{ТЧЭ}(d)) = 11$ *кОм*, при $I_0 = 0{,}1$ *мА* и $d = const$. Полученные результаты свидетельствуют о том, что материал ТЧЭ имеет свойство релаксации после воздействия. Фактор релаксации (абсолютное падение сопротивления рабочего тела ТЧЭ после напряжения и удержания мышц в напряженном состоянии) составляет: $k_p = 484$ *Ом/с*.

При расслаблении мышц также происходит резкое увеличение сопротивление материала ТЧЭ, но эта величина гораздо меньше относительно напряженного состояния ($R_{ТЧЭ}(d) = 6$ *кОм, при* $I_0 = 0{,}1$ *мА*). После резкого расслабления мышц материал ТЧЭ релаксирует со скоростью k_p до величины, равной 7 кОм (состояние отсутствия воздействия на ТЧЭ).

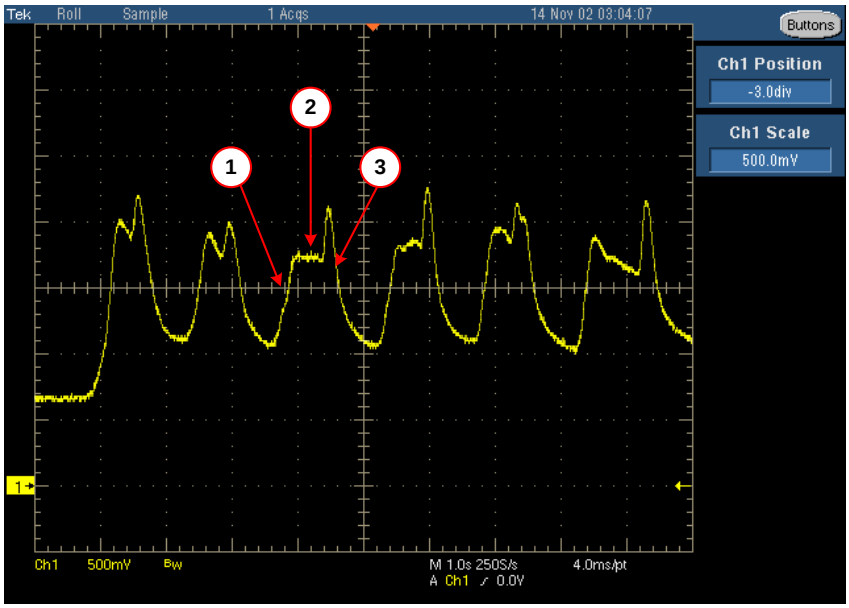


Рисунок 57 – Реакция ТЧЭ на серию воздействий напряжения и расслабления мышц

На рисунке 57 приведена реакция ТЧЭ на серию воздействий напряжения и расслабления мышц. Как видно из представленной зависимости, формы реакций на воздействия подобны, а различие заключается в крутизне и амплитуде фронта.

Величина фронта (1) (размах от "подошвы", предшествующей моменту напряжения мышц, до "пика", связанного с окончанием процесса напряжения мышц) пропорциональна напряжению мышц. "Полка сигнала" (2) отражает величину напряжения мышц. При расслаблении мышц, после всплеска,

возникающего в момент расслабления, происходит спад сигнала с одинаковой крутизной (3). Это говорит об одинаковом (полном) расслаблении мышц в каждом акте серии.

Из полученных результатов следует: явления напряжения и расслабления мышц характеризуются соответствующими неизменными формами сигнала, получаемого с ТЧЭ, что позволяет сделать вывод о возможности детектирования мышечного напряжения хвоста животного и, как результат, явлений, происходящих с хвостом животного.

5.3. Результаты исследований ЭТС при измерении мышечного напряжения на хвосте животного.

ЭТИП эксплуатировали в режиме измерения проводимости ТЧЭ с частотой 16 Гц. Сопоставление видеозаписи и данных, полученных с ЭТИП, показало полную адекватность фиксируемых явлений.

Анализ пиков, провалов, углов наклона графика, построенного на основании полученных экспериментальных данных, представленных в виде временной зависимости, позволяет однозначно, с несущественной задержкой, вызванной временем реакции оборудования и математической обработкой данных, определить время начала и окончания процесса дефекации и уринации.

На рисунке 58 представлена временная диаграмма с зафиксированными явлениями. По оси абсцисс отложены номера выборок, по оси ординат отложены значения данных, нормированные на максимальное значение.

Неоднородность среднего значения в центре обусловлена соскальзыванием ЭТИП с корня хвоста до момента его повторной фиксации.

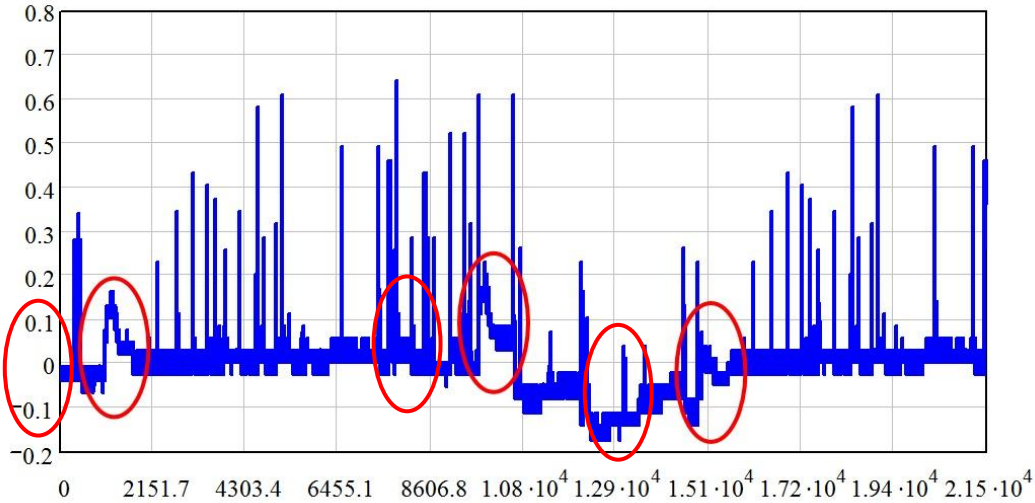


Рисунок 58 − Временная диаграмма с зафиксированными процессами дефекации и уринации (выделено красным). Слева направо: два явления дефекации, явление уринации

Из рисунка 58 видно, что явления дефекации и уринации четко различимы на временной диаграмме (временной масштаб изменен). На рисунке 59 приведен результат применения одного из возможных вариантов математической модели автоматического обнаружения явлений, происходящих с хвостом животного к случаю, представленному на рисунке 58.

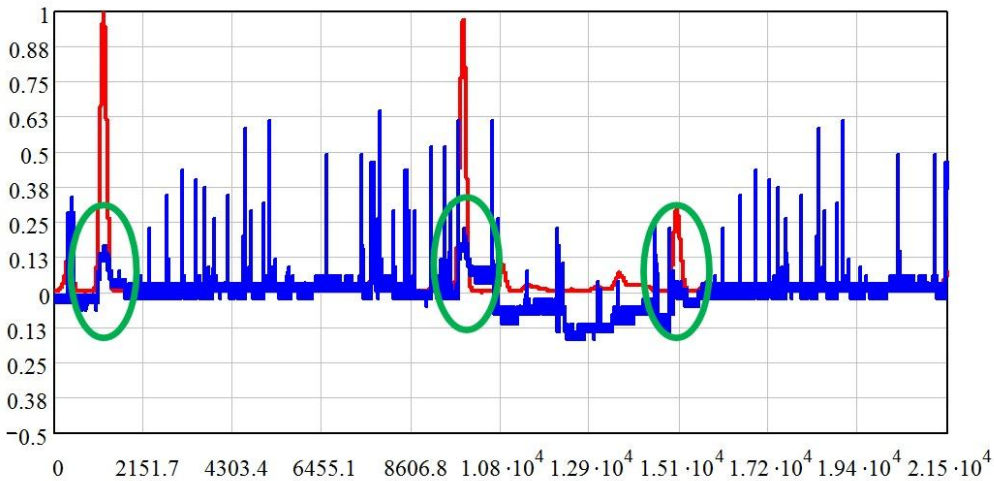


Рисунок 59 – Временная диаграмма с зафиксированными процессами дефекации и уринации (выделено зеленым). Слева направо: два явления дефекации, явление уринации. Синяя временная диаграмма − поток данных, поступающих на вход математической модели автоматического обнаружения явлений, происходящих с хвостом животного. Красная временная диаграмма − поток данных на выходе математической модели

Хорошо видно, что ЭТИП и математическая модель надежно и своевременно детектируют явление дефекации и уринации, а явления, не представляющие интерес (корова машет хвостом, отгоняет насекомых и т.п.), – игнорирует.

5.4. Результаты исследований ЭТС при родах животного.

ЭТИП эксплуатировали в режиме измерения проводимости ТЧЭ с частотой 16 Гц. Исследования показали, что ЭТС работоспособно в условиях фермы и обеспечивает достоверную регистрацию начала родов у коровы. Сопоставление видеозаписи и данных, полученных с ЭТИП, показало полную адекватность фиксируемых явлений.

Анализ пиков, провалов, углов наклона графика, построенного на основании полученных экспериментальных данных и представленных в виде временной зависимости, позволяет однозначно, с несущественной задержкой, вызванной временем реакции оборудования и математической обработкой данных, определить время начала и окончания схваток животного во время отела.

На рисунке 59 представлена временная диаграмма с зафиксированными явлениями. По оси абсцисс отложены номера выборок, по оси ординат отложены значения данных, нормированные на максимальное значение.

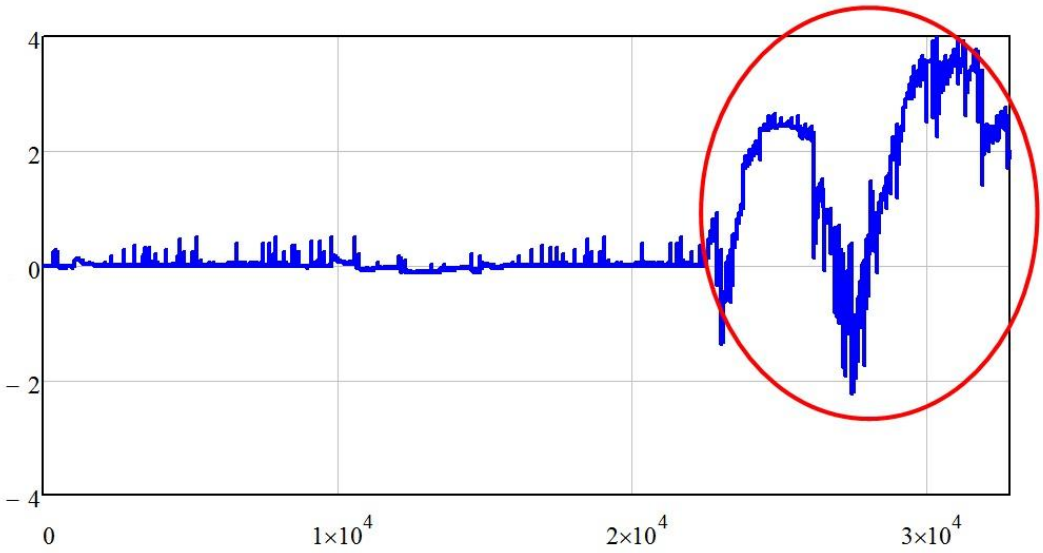


Рисунок 60 − Временная диаграмма с зафиксированными явлениями схваток (выделено красным)

Из рисунка 60 видно, что явления схваток четко различимы на временной диаграмме (временной масштаб изменен). На рисунке 60 приведен результат применения одного из возможных вариантов математической модели автоматического обнаружения явлений, происходящих с хвостом животного к случаю, представленному на рисунке 60. Так же в поток данных со схватками искусственно внедрены данные с рисунка 58 с явлениями дефекации и уринации (красная пунктирная линия), чтобы наглядно продемонстрировать работу СЦФ, настроенного на определение схваток.

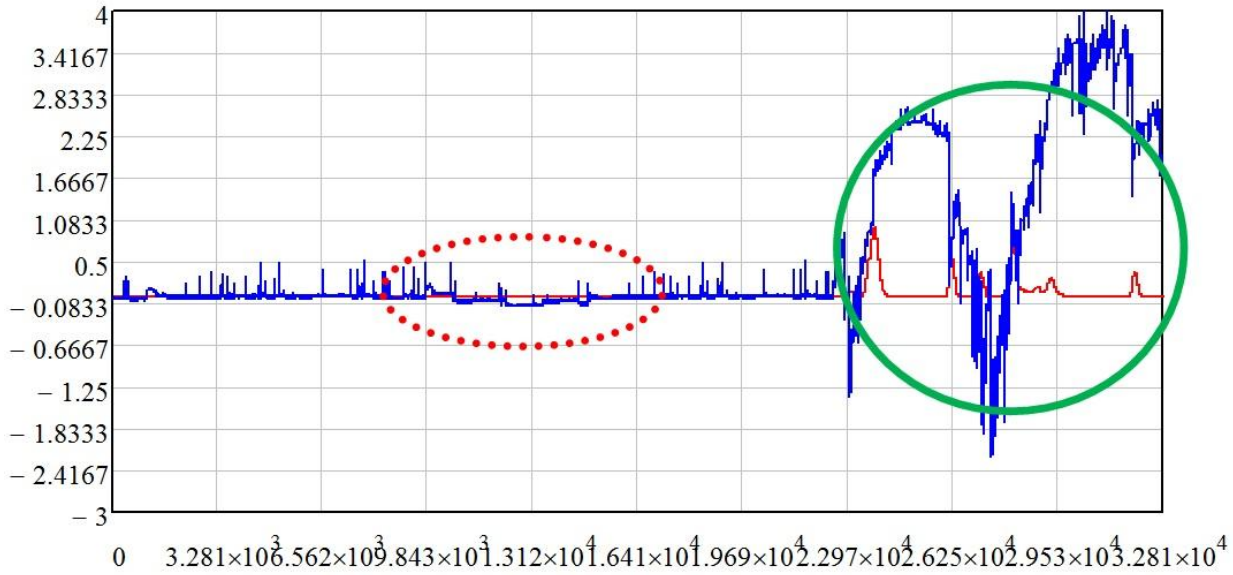


Рисунок 61 – Временная диаграмма с зафиксированными явлением схваток (выделено зеленым). Синяя временная диаграмма − поток данных, поступающих на вход математической модели автоматического обнаружения явлений, происходящих с хвостом животного. Красная временная диаграмма − поток данных на выходе математической модели. Красным пунктиром выделена область с искусственно внедренными явлениями дефекации и уринации

Хорошо видно, что ЭТИП, созданный на основе разработанной математической модели, надежно и своевременно детектирует явления родов, а явления, не представляющие интерес (дефекация, уринация, махание хвостом и т.п.), – игнорирует.

ГЛАВА 6. ДОСТОВЕРНОСТЬ ДЕТЕКТИРОВАНИЯ РОДОВЫХ СХВАТОК И АППАРАТНАЯ НАДЕЖНОСТЬ ЭЛЕКТРОТЕХНИЧЕСКИХ СРЕДСТВ МОНИТОРИНГА НАЧАЛА ОТЕЛА У КОРОВ.

Достоверность детектирования родовых схваток электротехническими средствами мониторинга технически сводится к решению задачи надежной интерпретации сигналов, поступающих с ЭТИП.

При решении этой задачи возможны два класса ошибок:

1 − «обнаружение» родовых схваток тогда, когда родовых схваток не было;

2 − необнаружение родовых схваток тогда, когда родовые схватки были.

Возникновение ошибок необнаружения вызывает более негативные последствия, поэтому недопустимы.

Уменьшить вероятность ошибок второго класса можно двумя способами: 1 − увеличить чувствительность ЭТИП, расположенного на корне хвоста; 2 − уменьшить избирательность цифровых фильтров СПО, то есть, огрубив интерпретацию сигнала.

Однако такой подход к решению возникшей проблемы имеет то негативное последствие, что за сигналы родовых схваток будут восприниматься побочные сигналы (шумы и помехи), вследствие чего возникнет излишняя и неоправданная нагрузка на персонал. То есть, вместо того, чтобы облегчить работу персонала и сделать ее более качественной, система будет выполнять обратную функцию, что так же недопустимо.

Поскольку никакой иной информацией, подлежащей интерпретации, кроме поступающей на вход микроконтроллера от ЭТИП, СПО не располагает, то для радикального обеспечения надежной работы системы необходимо подключение дополнительного опорного ЭТИП родовых схваток, функционирующего по совершенно иному принципу, чем базовый ЭТИП. Требование к различным физическим принципам функционирования ЭТИП является определяющим, поскольку только в этом случае интерпретация коррелирующих сигналов от двух

различных ЭТИП в реальном масштабе времени радикально снизит вероятность возникновения ошибки как первого, так и второго вида.

В качестве опорного сигнала, и, соответственно, опорного ЭТИП, использовался сигнал биопотенциала, а именно потенциала действия (ПД), вызывающий сокращение мышц, которое, в свою очередь, и является родовыми схватками. По сути, опорный ЭТИП представляет собой известный прибор – электромиограф, подключенный ко входу АЦП микроконтроллера через встроенный аналоговый мультиплексор, и расположенный на теле животного. Поскольку размах сигнала ПД составляет порядка нескольких десятков мВ (рисунок 62), то для его усиления применялся операционный усилитель (ОУ), также входящий в состав микроконтроллера.

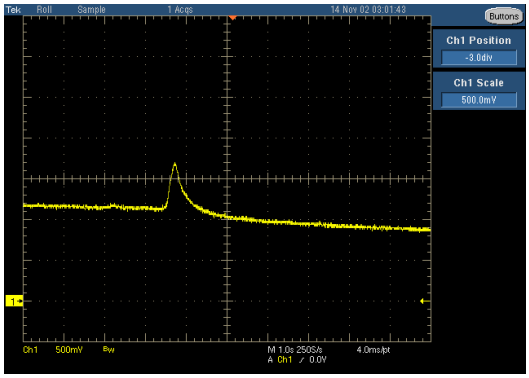


Рисунок 62 – Осциллограмма ПД

Таким образом, добавление канала опорного сигнала ПД, радикально улучшающего основные параметры системы, не приводит к усложнению архитектуры устройства, а требует лишь расширения возможностей СПО, функционирующей на ПЭВМ. Схема электрическая структурная канала опорного сигнала представлена на рисунке 63.

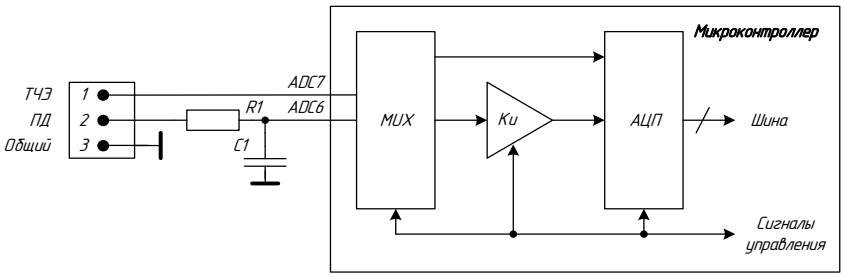


Рисунок 63 – Схема электрическая структурная канала опорного сигнала

Вход ОУ представляет собой фильтр верхних частот первого порядка, состоящий из конденсатора C1 (2,2 нФ) и резистора R1 (180 кОм). Частота среза входного ФВЧ близка к 400 Гц. Фильтр необходим для подавления постоянной составляющей сигнала, вызванной электрогальваническими явлениями на границе раздела электрод−кожа животного, а также подавления помехи тока промышленной частоты 50 Гц в 8 раз. Верхняя граница спектра сигнала ПД простирается до 1000−10000 Гц, поэтому фронт сигнала ПД проходит через ФВЧ практически без подавления. Далее отфильтрованный сигнал ПД поступает на вход усилителя с коэффициентом усиления Ku=200. С выхода усилителя сигнал поступает на вход АЦП (рисунок 63).

Исходя из эмпирически полученных величин сигналов родового акта и помех (рисунок 61), можно полагать, что вероятность возникновения ошибки первого класса составит 10^{-5}. То есть персонал будет ложно оповещен о наличии родовых схваток один раз на 100000 случаев, что, в случае нахождения в родильном отделении 100 голов, составит не чаще, чем один раз в год. Ошибок второго класса, которые приводят к более негативным последствиям вплоть до гибели коровы и теленка, будет значительно меньше, так как родовые схватки возникают не один раз, и если в первый раз детектирование родовых схваток ошибочно не произойдет с вероятностью 10^{-5}, то во второй раз произойдет с вероятностью $1-10^{-5}$. Таким образом, вероятность пропуска серии из двух родовых схваток составит $10^{-5} \cdot 10^{-5} = 10^{-10}$. А поскольку родовые схватки происходят не один и не два раза, а больше, то пропуск родового акта системой практически невозможен.

Однако, помимо надежности алгоритмов работы системы, остается вопрос аппаратной надежности элементов системы, которая складывается из надежности работы ПЭВМ и надежности работы периферийных устройств. Радикальное увеличение аппаратной надежности элементов системы достигается путем мониторинга работы элементов системы в реальном масштабе времени, и периодического информирования о текущей работоспособности системы. Таким образом, организация работы системы по известным алгоритмам может

обеспечить сколь угодно высокую защиту от сбоя функционирования аппаратных средств.

Система построена так, что достоверность детектирования родовых схваток и аппаратная надежность системы сопоставимы и высоки.

ГЛАВА 7. ЭКОНОМИЧЕСКАЯ ЭФФЕКТИВНОСТЬ ПРИМЕНЕНИЯ ЭТС ДЛЯ МОНИТОРИНГА НАЧАЛА ОТЕЛА У КОРОВ.

7.1. Факторы экономической эффективности применения ЭТС для определения отела у коров.

В экономической части рассчитывается экономическая эффективность применения ЭТС для определения отела у коров.

Для расчета экономической эффективности выделяются факторы эффективности применения электротехнической системы для своевременного оповещения о начале отела коров:

- Снижение смертности коров;
- Снижение смертности телят;
- Снижение затрат на лечение коров и телят из−за проблем с их здоровьем, возникающих в результате неоказания помощи при отеле;
- Снижение затрат на воспроизводство стада из−за снижения смертности коровы и теленка. В случае смерти коровы потребуется закупка нетели или ее воспроизводство на ферме, что требует наличия дополнительно количества скотомест;
- Увеличение производства молока по ферме, т.к. здоровая, отелившаяся корова сразу начинает давать молоко. Погибшая − требует покупки нетели, которую до отела необходимо несколько месяцев содержать, а заболевшая снизит надои.

Для расчетов примем, что разработанная система применяется на молочной ферме на 100 дойных коров, полагая, что эффективность данной системы будет возрастать при увеличении размера фермы (т.е. количества на ней коров).

Введем понятие экономического ущерба. Экономический ущерб – это выручка фермы, которая могла бы быть получена, если бы удалось устранить дистоцию. Т.е. экономический ущерб является недополученным доходом. При

внедрении разработанной ЭТС на ферму экономический ущерб от дистоции возможно свести к минимуму.

На основе вышеприведенных факторов выделяются три основных вида экономического ущерба, наносимого ферме вследствие дистоции (рисунок 64):

- Экономический ущерб от выбытия нетели;
- Экономический ущерб от потери молока нерожденных телят (женского пола);
- Экономический ущерб от потери мяса нерожденных телят.

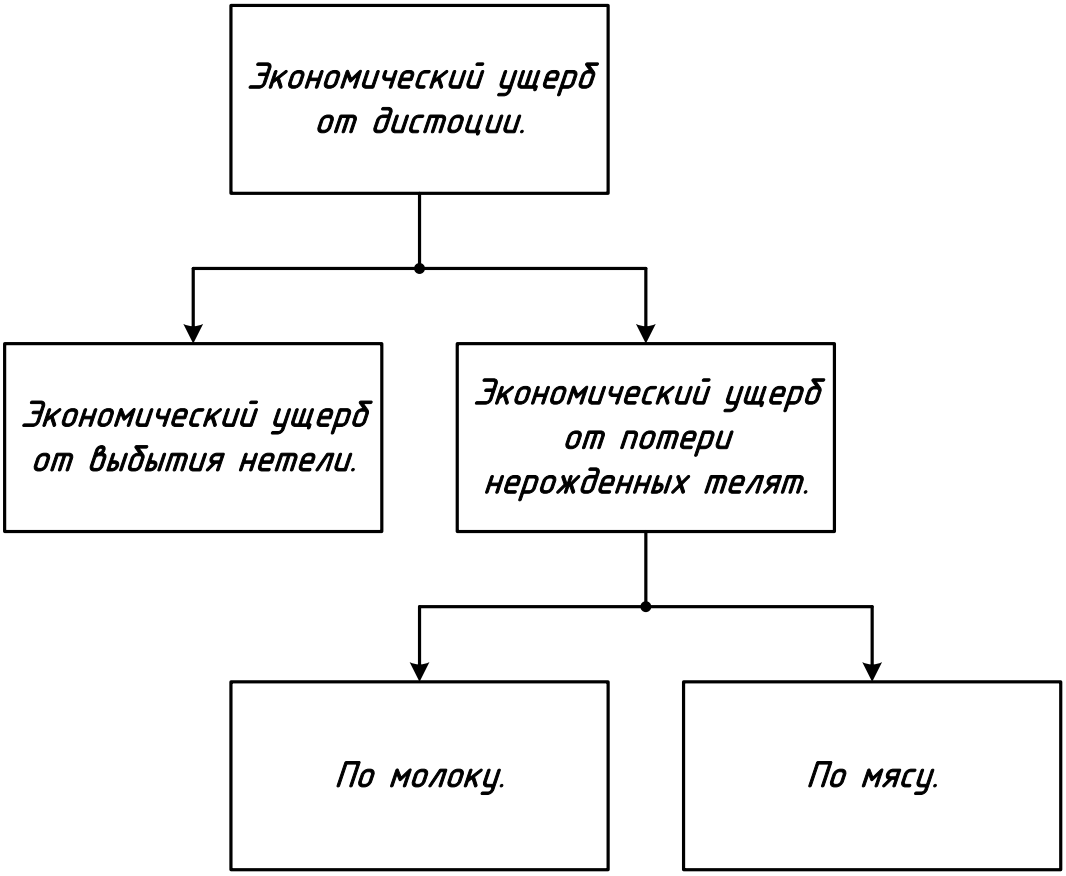


Рисунок 64 – Экономический ущерб от дистоции коров

Экономический ущерб (издержки) от выбытия нетели, в свою очередь, состоит из альтернативной выручки от реализации продукции, производимой коровой, и стоимости покупки новой нетели. Изобразим это схематически на рисунке 65:

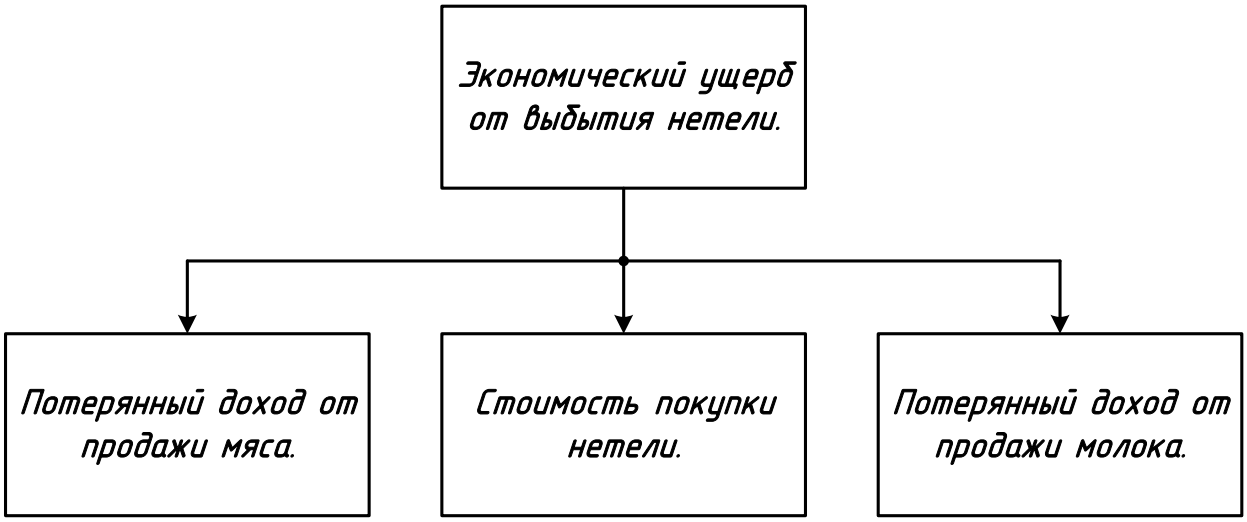


Рисунок 65 – Экономический ущерб от выбытия нетели

Тогда будем рассчитывать экономический ущерб от выбытия нетели ($У_{нетели}$) по следующему выражению:

$$У_{нетели}=Э_{мяса}+Э_{молока}+Ц_{нетели}, \quad (54)$$

где:

- $Э_{мяса}$ – экономические издержки от продажи мяса;
- $Э_{молока}$ – экономические издержки от продажи молока;
- $Ц_{нетели}$ – цена новой нетели;

В свою очередь, экономические издержки от продажи молока и мяса будем вычислять следующим образом:

$$Э_{мяса}=М_{н}\cdot Ц_{мяса}\cdot В_{убой}, \quad (55)$$

где:

- $М_{н}$– средняя масса нетели (кг);
- $Ц_{мяса}$ – закупочная цена за 1 кг говядины (руб.),
- $В_{убой}$ – доля убойного выхода.

$$Э_{молока}=Q_{молока}\cdot Ч_{отелов}\cdot Ц_{молока}, \quad (56)$$

где:

- $Q_{молока}$ – удой молока одной коровы за лактацию (кг);
- $Ч_{отелов}$ – среднее число отелов одной коровы за жизнь;
- $Ц_{молока}$ – закупочная цена за 1 кг молока (руб.);

Экономический ущерб от потери нерожденного теленка по молоку ($У_{молока}$) будем рассчитывать с учетом того, что вероятность появления теленка мужского или женского пола равна 50:50.

$$У_{молока}=Э_{молока}\cdot р_{пола}, \quad (57)$$

где:

- $p_{пола}$ – вероятность появления теленка женского пола;

Экономический ущерб от потери нерожденного теленка по мясу ($У_{мяса}$) так же будем рассчитывать с учетом того, что вероятность появления теленка мужского или женского пола равна 50:50, а масса коровы и быка сильно различается. Тогда получается следующее выражение:

$$У_{мяса}=Ц_{мяса}\cdot[М_к\cdot р_{пола}+М_б\cdot(1-р_{пола})], \quad (58)$$

где:

- $M_к$ – средняя масса коровы (кг);
- $M_б$ – средняя масса быка (кг);

Тогда экономический ущерб от дистоции коровы будет рассчитан как:

$$Э_{дистоции}=У_{нетели}+Ч_{отелов}\cdot(У_{мяса}+У_{молока}). \quad (59)$$

На основе этого показателя мы сможем рассчитать общий экономический ущерб, который наносится всей ферме в результате дистоции коров в будущем периоде:

$$Э_{общий}=Ч_{коров}\cdot р_{смерти}\cdot Э_{дистоции}, \quad (60)$$

где:

- $p_{смерти}$ – вероятность смерти коровы от дистоции;

- $Ч_{коров}$ – численность коров на ферме;

Для расчета фактического экономического ущерба, наносимого ферме в течение года, используется выражение:

$$Э_{факт}=(Ц_{нетели}+Q_{молока}\cdot Ц_{молока})\cdot Ч_{коров}\cdot p_{смерти}. \quad (61)$$

7.2. Расчет стоимости ЭТС и определение затрат на его обслуживание.

Приведем стоимость необходимых для производства ЭТС комплектующих в таблицах 16…18.

Таблица 16 – Стоимость комплектующих устройства сопряжения с ПЭВМ

№ п/п	Наименование	шт.	Цена (руб.) за шт.	Цена (руб.)
1	Конденсаторы GCM155R71C104KA55D “Murata”	3	2,34	7,02
2	Микросхема ATmega16−16MU “Atmel”	1	1,5	1,5
3	Микросхема ADM485EARZ “Analog Devices”	1	6,85	6,85
4	Резистор RG1005P−100−B−T5 “Susumu”	1	1,5	1,5
5	Резистор RG1005P−50−B−T5 “Susumu”	1	1,5	1,5
6	Резистор RG1005P−4,73−B−T5 “Susumu”	1	1,5	1,5
7	Резистор RG1005P−120−B−T5 “Susumu”	1	1,5	1,5
8	Резистор RC0402JR−0722R “Yageo”	1	1,5	1,5
9	Светодиод KPHHS−1005CGCK “Kingbright”	1	6,52	6,52
10	M50−3600542 “Harwin”	1	77,1	77,1
11	Розетка PBD16 “Connfly Electronic”	1	6,3	6,3
12	Печатная плата	1	112,52	112,52
Итого устройство сопряжения:				225,31

Таблица 17 – Стоимость комплектующих устройства ЭТИП

№ п/п	Наименование	шт.	Цена (руб.) за шт.	Цена (руб.)
1	Конденсаторы GCM155R71C104KA55D “Murata”	2	2,34	4,68
2	Микросхема ATmega16L−8MU “Atmel”	1	1,43	1,43
3	Индуктивность CM322522−100KL “Bourns”	1	4,82	4,82
4	Резистор RC0402JR−0722R “Yageo”	1	1,5	1,5
5	Резистор RG1005P−4,73−B−T5 “Susumu”	1	1,5	1,5
6	Резистор RG1005P−103−B−T5 “Susumu”	1	1,5	1,5
7	Резистор RG1005P−223−B−T5 “Susumu”	1	1,5	1,5
8	Резистор RG1005P−393−B−T5 “Susumu”	1	1,5	1,5
9	Резистор RG1005P−823−B−T5 “Susumu”	1	1,5	1,5
10	Светодиод KPHHS−1005CGCK “Kingbright”	1	6,52	6,52
11	M50−3600542 “Harwin”	1	77,1	77,1
12	M50−3600542 “Harwin”	1	77,1	77,1
13	Розетка PBD16 “Connfly Electronic”	1	6,3	6,3
14	Аккумулятор	1	102,51	102,51
15	Печатная плата	1	112,52	112,52
Итого устройство ЭТИП:				401,98

Таблица 18 – Общая стоимость материалов

№ п/п	Наименование	шт.	Цена (руб.) за шт.	Цена (руб.)
1	Итого устройство сопряжения:	1	225,31	225,31
2	Итого устройство ЭТИП:	1	401,98	401,98
3	Корпус для РЭА	2	150	300,00
	Итого стоимость материалов:			927,29

Для определения себестоимости 1 шт. выпускаемого ЭТС рассчитаем стоимость сборки и изготовления при заказе на предприятии ООО «Микролит». Планируемый годовой объем производства составит 100000 шт. Тогда подготовка производства будет стоить 11000 руб., а стоимость монтажа поверхностных компонентов при автоматическом монтаже составит 15 руб. на единицу продукции. В этом случае подготовка производства в расчете на единицу

продукции составит 0,11 руб., тогда итоговая стоимость монтажа за 1 шт. будет равна 15,11 руб.

Прочие расходы включают в себя корпусирование и упаковку изделия, по нашим расчетам они будут составлять 70% от стоимости основных материалов и монтажа, т.е. 659,68 руб.

Произведем расчет себестоимости единицы продукции (таблица 19). Она будет включать в себя стоимость материалов, стоимость сборки и изготовления на стороннем предприятии и прочие расходы.

Таблица 19 – Себестоимость единицы продукции

Наименование	**шт.**	**Цена (руб.) за шт.**	**Цена (руб.)**
Цена монтажа поверхностных компонентов (автоматический монтаж)	100	0,15	15
Подготовка производства (за ед.)	100000	11000	0,11
Итого монтаж:			15,11
Прочие расходы:			659,68
Итого стоимость материалов:			927,29.
Себестоимость:	1	1602,08	1602,08

Для определения отпускной цены ($Ц_О$) заложим в получившуюся себестоимость норму прибыли в 20% и НДС 18%:

$$Ц_О = 1{,}18 \cdot 1{,}2 \cdot 1602{,}08 \text{ руб.} = 2268{,}55 \text{ руб.}$$

Капиталовложения проектного варианта с учетом монтажа и накладных расходов составят:

$$К_п = Ц_о + М_н + Н_р, \quad (62)$$

где:

- $Ц_о$ – цена устройства;
- $М_н = 0{,}1\ Ц_о$ – монтажные работы;
- $Н_р = 0{,}125\ Ц_о$ – накладные расходы.

Цена устройства составляет 2268,55 руб.

$М_н$ = 0,1 $Ц_о$= 0,1*2268,55 руб. = 226,86 руб.

$Н_р$ = 0,125*2268,55 руб. = 283,57 руб.

$К_п$=2268,55 руб. + 226,86 руб. + 283,57 руб. = 2778,97 руб.

Затраты на эксплуатацию устройства рассчитываются следующим образом:

$$З_{э_п}=А_о+Р_{то}+С_{ээ}+П_о, \quad (63)$$

где:

- A_o – амортизационные отчисления;
- $P_{то}$ – затраты на ремонт и техническое обслуживание;
- $C_{ээ}$ – стоимость потребляемой электроэнергии;
- $П_o$ – прочие расходы.

$$А_0 = К_П \cdot \frac{а}{100\%}, \quad (64)$$

где а – годовая норма амортизационных отчислений, %.

$$Р_{то} = К_П \cdot \frac{ч}{100\%}, \quad (65)$$

где ч – годовая норма отчисления на ремонт и техобслуживание, %.

Стоимость потребляемой энергии рассчитывается по следующей формуле:

$$C_{ээ} = Q_{ээ} \cdot T_{ээ}, \quad (66)$$

где:

- $Q_{ээ}$ – расход потребляемой электроэнергии в год, кВт/ч;
- $T_{ээ}$ – тариф на электроэнергию, руб./(кВт/ч).

$П_o = 0{,}01 \cdot K_п$.

$А_0 = 2778{,}97$ руб.$\cdot \frac{14{,}2\%}{100\%}$ = 394,61 руб.

$Р_{то} = 2778{,}97$ руб.$\cdot \frac{5{,}6\%}{100\%}$ = 155,62 руб.

$С_{ээ} = 10{,}8 кВт/ч \cdot 4{,}16\ руб./(кВт/ч)$ = 44,93 руб.

$П_о = 0{,}01 \cdot 2778{,}97\ руб.$ = 27,79 руб.

$З_{э_п} = 394{,}61\ руб. + 155{,}62\ руб. + 44{,}93\ руб. + 27{,}79\ руб.$ = 622,95 руб.

Тогда затраты по приобретению одного ЭТС составят:

$$З_{этс} = З_{э_п} + К_п = 622{,}95 \text{ руб.} + 2778{,}97 \text{ руб.} = 3401{,}93 \text{ руб.}$$

7.3. Расчет экономической эффективности применения ЭТС.

Показатели для расчета экономического ущерба от дистоциии и их значения приведены в таблице 20:

Таблица 20 – Показатели для расчета экономического ущерба от дистоции

№ п/п	Показатель	Условное обозначение	Значение
1	Среднее число отелов одной коровы за жизнь	$Ч_{отелов}$	3
2	Вероятность появление теленка женского пола	$р_{пола}$	0,5
3	Численность коров на ферме	$Ч_{коров}$	100
4	Вероятность смерти коровы от дистоции	$р_{смерти}$	0,02
5	Средняя масса нетели (кг)	$М_{н}$	450
6	Доля убойного выхода	$В_{убой}$	0,5
7	Удой молока одной коровы за лактацию (кг)	$Q_{молока}$	7 000
8	Закупочная цена 1 кг говядины (руб.)	$Ц_{мяса}$	170
9	Закупочная цена 1 кг молока (руб.)	$Ц_{молока}$	20
10	Цена новой нетели (руб.)	$Ц_{нетели}$	100 000
11	Средняя масса коровы (кг)	$М_{к}$	600
12	Средняя масса быка (кг)	$М_{б}$	800

$Э_{мяса}=М_{н}\cdot Ц_{мяса}\cdot В_{убой}=450\cdot 170\cdot 0{,}5=38250$ руб.

$Э_{молока}=Q_{молока}\cdot Ч_{отелов}\cdot Ц_{молока}=7000\cdot 3\cdot 20=420000$ руб.

$У_{нетели}=Э_{мяса}+Э_{молока}+Ц_{нетели}=38250+420000+100000=558250$ руб.

$У_{молока}=Э_{молока}\cdot р_{пола}=420000\cdot 0{,}5=210000$ руб.

$У_{мяса}=Ц_{мяса}\cdot(М_{к}\cdot р_{пола}+М_{б}\cdot(1-р_{пола}))=170\cdot(600\cdot 0{,}5+800\cdot(1-0{,}5))=119000$ руб.

$Э_{дистоции}=У_{нетели}+Ч_{отелов}\cdot(У_{мяса}+У_{молока})=558250+3\cdot(119000+210000)=1545250$ руб.

$Э_{общий}=Ч_{коров}\cdot р_{смерти}\cdot Э_{дистоции}=3090500$ руб.

Значит, потерянный доход фермы от дистоции в будущих периодах составляет 3 090 500 руб.

Рассчитаем фактический экономический ущерб, наносимый в результате дистоции за год:

$Э_{факт}=(Ц_{нетели}+Q_{молока}\cdot Ц_{молока})\cdot Ч_{коров}\cdot р_{смерти}=(100000+7000\cdot 20)\cdot 100\cdot 0{,}02=480$ 000 руб.

Для дальнейшей оценки экономической эффективности возьмем данные по финансовым результатам для молочной фермы на 100 голов, рассчитанные Минсельхозом РФ [56]. Выручка фермы от продажи (7,0 млн. руб.) будет формироваться за счет поставки сырого молока на переработку (6,1 млн. руб. в год), а также от продажи мяса телят, полученных от приплода (480 тыс. руб. в год), и мяса выбракованных коров (500 тыс. руб.).

Ежегодные операционные расходы фермы планируются в размере 5,9 млн. руб., из которых 4,0 млн. руб.— расходы на корма, 1,0 млн. руб. — расходы на оплату труда и 0,9 млн. руб. — эксплуатационные и иные расходы. Операционная прибыль фермы при этом составит 1,1 млн. руб.

Прибыль ферм формируется не только за счет производства молока, но и за счет участия в перерабатывающем потребительском кооперативе, выручка от переработки молока составляет 10,45 млн. руб. Расходы по переработке на ферму составляют 7,1 млн. руб. Таким образом, операционная прибыль от переработки молока равняется 3,35 млн. руб.

Тогда общая выручка фермы составит 17,45 млн.руб., а общие расходы – 13 млн.руб. Общая операционная прибыль фермы (П) составит 4,45 млн.руб.

Будем считать, что применение ЭТС снизит вероятность негативных последствий дистоции на 80% за счет появления возможности своевременного оказания помощи корове при родах. Поэтому для расчета выручки при внедрении ЭТС увеличим размер выручки от продаж на величину, равную 80% общего экономического ущерба от дистоции, т.к. он является фактически недополученным доходом.

При этом цена одного ЭТС равна 2268,55 руб., будем считать, что на 100 голов, в пиковый период, требуется 25 приборов. Тогда затраты фермы на приобретение комплекта ЭТС (З) составят 85048 руб. Посчитаем изменение расходов и выручки при внедрении ЭТС:

$B_э$=В+0,8·$Э_{факт}$=17834000 руб.

$P_э$=Р+З=13000000+85048=13085048,21 руб.

Тогда операционная прибыль составит:

$П_э = В_э - Р_э$=4748951,79 руб.

Рассчитаем коммерческий эффект – изменение прибыли в абсолютном выражении ($Э_{фх}$):

$Э_{фх} = П_э - П$ =4748951,79–4450000=298951,79 руб.

Исходя из этого, посчитаем изменение прибыли в относительном выражении:

$Т = \frac{Э_{фх}}{П} \cdot 100\% = 6{,}72\%$

Сделаем вывод, что за счет уменьшения общего экономического ущерба от дистоции при внедрении разработанной нами ЭТС (при прочих равных условиях) операционная прибыль молочной фермы на 100 голов может возрасти примерно на 6,72%.

Теперь рассчитаем срок окупаемости капиталовложений как отношение капиталовложений к экономическому эффекту:

$$Т_{ок} = \frac{К_п}{Э_{фх}} = 0{,}23\ года$$

Тогда срок окупаемости в днях составит 85 дней.

Экономическая эффективность капиталовложений является обратной величиной сроку окупаемости и рассчитывается как отношение экономического эффекта к стоимости капиталовложений:

$$Е_Ф = \frac{Э_{фх}}{К_п} = 4{,}30$$

Сравним получившееся значение с $Е_н$ – нормативным коэффициентом эффективности капитальных вложений, равным 0,25. Видим, что $Е_ф \geq Е_н$, значит, наш проект является экономически эффективным.

Теперь рассчитаем показатели рентабельности, которые являются обобщающим итоговым показателем экономической эффективности. На основе имеющихся данных мы можем рассчитать рентабельность издержек, рентабельность продаж и рентабельность инвестиций. Первый показатель характеризует выход прибыли в процессе реализации продукции на единицу издержек в основной деятельности. Второй показатель − рентабельность продаж −

характеризует доходность основной деятельности фермы. Третий – характеризует экономическую эффективность осуществленного капиталовложения.

Сначала рассчитаем показатели рентабельности фермы до внедрения ЭТС. Рентабельность издержек рассчитывается как отношение операционной прибыли (П) к операционным расходам (Р):

$$R_{издержек}=\frac{П}{Р}\cdot 100\%=34{,}23\%$$

Рентабельность продаж рассчитывается как отношение операционной прибыли (П) к выручке от продаж (В):

$$R_{продаж}=\frac{П}{В}\cdot 100\%=25{,}50\%$$

Теперь рассчитаем эти же показатели рентабельности, но с учетом изменения данных в связи с внедрением ЭТС. Тогда получим следующие значения рентабельности:

$$R_{издержек}=\frac{П_э}{Р_э}\cdot 100\%=36{,}29\%$$

$$R_{продаж}=\frac{П_э}{В_э}\cdot 100\%=26{,}63\%.$$

Мы видим, что рентабельность издержек увеличилась на 2,06% в абсолютном выражении, или на 6,02% в относительном. При этом рентабельность продаж увеличилась на 1,13% в абсолютном выражении, или на 4,42% в относительном выражении, что свидетельствует о значительном увеличении экономической эффективности фермы.

В результате проведения расчетных мероприятий мы выяснили, что при внедрении разработанного ЭТС на молочной ферме на 100 голов дойного стада коммерческий эффект составит 298951,79 руб. Тогда следует ожидать прироста прибыли более чем на 6%, прироста рентабельности продаж и издержек в относительном выражении на 4% и 6% соответственно.

Срок окупаемости капиталовложений составит 85 дней, рассчитанный коэффициент экономической эффективности в несколько раз превышает нормативное значение, − значит, внедрение ЭТС экономически эффективно.

Полученные результаты говорят об экономической целесообразности применения разработанного ЭТС.

ОСНОВНЫЕ ВЫВОДЫ

1. Разработан миографический метод мониторинга начала отела у коров, основанный на регистрации напряжения мышц корня хвоста, сопровождающего схватки и потуги животного, электротехническими средствами, обеспечивающий оповещение персонала о начале родовых схваток в режиме реального времени для своевременного родовспоможения, а также дополнительно контроль уринации, дефекации и отпугивания насекомых.

2. Разработана математическая модель измерительного преобразователя напряжения мышц корня хвоста животного в электрический сигнал, позволяющая реализовать измерительный преобразователь пригодный для взаимодействия с современными высокоточными АЦП (от 10 до 24 разрядов).

3. Разработан алгоритм обработки сигналов от измерительного преобразователя напряжения мышц корня хвоста животного для мониторинга начала отела у коров, позволяющий за пять шагов идентифицировать явления, происходящие с животным. Это достигается путем обработки низкочастотной и высокочастотной составляющих сигнала и усилением полезной составляющей, а так же использованием критерия χ−квадрат для сравнения гистограмм текущего состояния хвоста животного с гистограммами эталонных явлений. Для достоверного мониторинга предусмотрен синхронный анализ сигналов, поступающих от измерительного преобразователя электрического потенциала действия мышц.

4. Разработан электротехнический преобразователь напряжения мышц корня хвоста животного, принцип работы которого основан на измерении электропроводности рабочего тела, представляющего собой электролит − 10% раствор NaCl в воде, помещенного в гипоалергенную оболочку, обеспечивающий надежное регистрируемое изменение электропроводности в диапазоне 450−520 мкСм, пропорциональное площади сечения корня хвоста в плоскости, охватываемой преобразователем.

5. Разработан, изготовлен и прошел производственную проверку комплекс электротехнических и программных средств для мониторинга начала отела у

коров с дистанционной передачей данных персоналу фермы в режиме реального времени, в следующем составе: измерительный преобразователь напряжения мышц корня хвоста, измерительный преобразователь потенциала действия, устройство сопряжения с ПЭВМ, СПО для интерпретации полученных данных.

6. Проведены расчеты, показавшие высокую надежность разработанных ЭТС при среднем времени наработки на отказ, составляющем 1476470 час. Оригинальность технических и программных средств подтверждена патентом РФ на полезную модель № 134782 и свидетельством о регистрации программы для ЭВМ № 2016616713.

7. Оценка экономической эффективности применения комплекта электротехнических средств для мониторинга начала родов у коров показывает, что ожидаемый прирост прибыли молочной фермы на 100 голов дойного стада составляет более 6% при сроке окупаемости капиталовложений 85 дней, что обусловлено невысокой стоимостью ЭТС по отношению к стоимости возможных материальных потерь.

СПИСОК СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

AVR – семейство восьмибитных микроконтроллеров фирмы Atmel;

ISM – Industrial, scientific and medical;

JTAG – Joint Test Action Group;

MMS – Multimedia Messaging Service;

SMS – Short Messaging Service;

SPI – Serial Peripheral Interface;

UART – Universal Asynchronous Receiver−Transmitter. См. УАПП;

АЦП – аналого−цифровой преобразователь;

КИХ – конечная импульсная характеристика;

КРС − крупнорогатый скот;

МК – микроконтроллер;

МЭМС – микро электромеханическая система;

ОЗУ – оперативное запоминающее устройство;

ОС – операционная система;

ОСРВ – операционная система реального времени;

ПО – программное обеспечение;

ПЭВМ – персональная электронно−вычислительная машина;

РОН – регистр(ы) общего назначения;

СБИС – сверхбольшая интегральная схема;

СПО – специальное программное обеспечение;

СЦФ – специальный цифровой фильтр;

ТЧЭ – тензочувствительный элемент;

УАПП – универсальный асинхронный приемопередатчик;

ЦАП – цифро−аналоговый преобразователь;

ЭРИ – Электро−радио изделия;

ЭТС – электротехнические средства;

ЯВУ – язык высокого уровня;

ЯНУ – язык низкого уровня;

СПИСОК ЛИТЕРАТУРЫ

1. Абрашин А.А. Обоснование параметров электротехнической системы выявления половой охоты коров. / Абрашин А.А., Иванов Ю.Г. // Вестник ФГБОУ МГАУ. Сер. Агроинженерия. – 2012. – №2 – С. 23 – 27.

2. Автоматизация оборудования молочных ферм. / В. И. Сыроватка, Ю. А. Цой, А. И. Зеленцов. – М.: Росагропромиздат, 1989. − 262 с.

3. Автоматизация технологических процессов и теория автоматического управления / И. Ф. Бородин, С. А. Андреев. – М.: Колос, 2003. – 344 с.

4. Автоматизация технологических процессов. / И. Ф. Бородин, Ю. А. Судник. – М.: Колос, 2003. – 344 с.

5. Автоматика. / А. В. Шавров, А. П. Коломиец. – М.: Колос, 2000. – 182 с.

6. Адлер Ю. П. Введение в планирование эксперимента. – М.: Металлургия, 1969. – 159 с.

7. Анализ финансовой отчетности. / Е. Г. Жулина, Н. А. Иванова. − М.: Дашков и К, 2010. – 267 с.

8. Анализ хозяйственной деятельности предприятия. / В. В. Ковалев, О. Н. Волкова. – М.: Проспект, 2010. – 420 с.

9. Анго А. Математика для электро– и радиоинженеров. – М.: Наука, 1965. – 586 с.

10. Андреева Л. Е. Упругие элементы приборов. – М., Машгиз, 1962. – 456 с.

11. Вендин, С.В. К расчету конструктивных и режимных параметров разравнивающего устройства. / С.А. Булавин, С.В. Вендин, Ю.В. Саенко // Вестник Всероссийского научно−исследовательского института механизации животноводства. − 2015. − № 4 (20). − С. 149−152.

12. Вендин, С.В. К расчету напряженностей электромагнитного поля при СВЧ обработке диэлектрических плоскослоистых объектов. / С.В. Вендин, П.А. Трубаев // Вестник Белгородского государственного технологического университета им. В.Г. Шухова. − 2013. − № 6. − С. 215−218.

13. Вендин, С.В. К расчету распространения электромагнитного импульса при СВЧ обработке диэлектрических сред. / С.В. Вендин, И.А. Щербинин // Вестник

Белгородского государственного технологического университета им. В.Г. Шухова. − 2015. − № 2. − С. 204−206.

14. Вендин, С.В. К решению задачи взаимодействия электромагнитной волны с многослойным сферическим диэлектрическим объектом. / С.В. Вендин // Вестник Белгородского государственного технологического университета им. В.Г. Шухова. − 2013. − № 5. − С. 216−220.

15. Вендин, С.В. Стимулирующее влияние излучения свч−диапазона на сельскохозяйственных животных. / С.В. Вендин, В.В. Боцман, Г.С. Походня, Ю.В. Иванова, А.М. Стадник, Н.В. Черный // Вестник Курской государственной сельскохозяйственной академии. − 2014. − № 2. − С. 50−52.

16. Ветеринарное акушерство и гинекология. / А. П. Студенцов, В. С. Шипилов, Л. Г. Субботина, О. Н. Преображенский. – М.: Агропромиздат, 1986. – 480 с.

17. Ветеринарное акушерство, гинекология и биотехника размножения. / А. П. Студенцов, В. С. Шипилов, В. Я. Никитин. – М.: Колос, 1999. – 495 с.

18. Викторов А.И. Радиотехнический метод определения местонахождения животных и определения половой охоты коров и телок. / Викторов А.И., Иванов Ю.Г. // Механизация и электрификация сельского хозяйства. – 2005. – №4 – С. 9 – 11.

19. Вознесенский В. А. Статистические методы планирования эксперимента в технико–экономических исследованиях. М.: Статистика, 1974. – 263 с.

20. Газета "Хозяйство". Корову атакуют оводы и слепни. http://hozvo.ru/news/master_s_veterinarian/Korovuatakuyutovodyiislepni.htm.

21. ГОСТ 14254–96 Степени защиты, обеспечиваемые оболочками.

22. ГОСТ 15150–69 Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды.

23. ГОСТ 19.001–77 Единая система программной документации. Общие положения.

24. ГОСТ 19.701–90 Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
25. ГОСТ 2.104–2006 Единая система конструкторской документации. Основные надписи.
26. ГОСТ 2.105–95 Единая система конструкторской документации. Общие требования к текстовым документам.
27. ГОСТ 2.106–96 Текстовые документы.
28. ГОСТ 2.109–73 Основные требования к чертежам.
29. ГОСТ 2.413–72*(1–VII–84) Правила выполнения конструкторской документации изделий, изготавливаемых с применением электрического монтажа.
30. ГОСТ 2.702–75 Правила выполнения схем.
31. ГОСТ 2.708–81 Правила выполнения электрических схем цифровой и вычислительной техники
32. ГОСТ 2.709–89 Обозначения условные проводов и контактных соединений электрических элементов, оборудования и участков цепей в электрических схемах.
33. ГОСТ 2.710–81 Обозначения буквенно–цифровые в электрических схемах.
34. ГОСТ 2.721–74 ЕСКД. Обозначения условные графические в схемах. Обозначения общего применения.
35. ГОСТ 2.725–68 Единая система конструкторской документации. Обозначения условные графические в схемах. Устройства коммутирующие.
36. ГОСТ 2.728–74 Единая система конструкторской документации. Обозначения условные графические в схемах. Резисторы, конденсаторы.
37. ГОСТ 2.730–73 Обозначения условные графические в схемах. Приборы полупроводниковые.
38. ГОСТ 2.743–91 Единая система конструкторской документации. Обозначения условные графические в схемах. Элементы цифровой техники.

39. ГОСТ 2.747–68 Обозначения условные графические в схемах. Размеры условных графических обозначений.
40. ГОСТ 2.755–87 Обозначения условные графические в электрических схемах. Устройства коммутационные и контактные соединения.
41. ГОСТ 2.756–76 Обозначения условные графические в схемах. Воспринимающая часть электромеханических устройств.
42. Гришин, И.И. Измерения электрофизических параметров биоткани. / И.И. Гришин, В.В. Лапутин // Труды международной научно−технической конференции Энергообеспечение и энергосбережение в сельском хозяйстве. − 2012. Т. 5. − С. 163−168.
43. Гришин, И.И. Математическое обоснование напряженности электрического поля в вымени коров от четырех электродов. / И.И. Гришин, А.С. Морозов, Е.С. Семина // Труды международной научно−технической конференции Энергообеспечение и энергосбережение в сельском хозяйстве. − 2012. Т. 1. − С. 365−370.
44. Гришин, И.И. Методы измерения комплексной проводимости живых тканей. / И.И. Гришин, В.В. Третьякова // В сборнике: Актуальные проблемы и их инновационные решения в АПК. Сборник научных трудов. Посвящается 60−летию инженерного факультета. Рязанский государственный агротехнологический университет имени П.А. Костычева. Рязань, − 2011. − С. 16−19.
45. Гришин, И.И. Методы измерения комплексной проводимости живых тканей. / И.И. Гришин, В.В. Третьякова // В сборнике: Актуальные проблемы и их инновационные решения в АПК. Сборник научных трудов. Посвящается 60−летию инженерного факультета. Рязанский государственный агротехнологический университет имени П.А. Костычева. Рязань, − 2011. − С. 16−19.
46. Гулсен. Я. Сигналы коров. – Нидерланды, 2010. – 98 с.
47. Документация «AVR Instruction set».
48. Документация на микроконтроллер «ATmega16».

49. Документация на микросхему без проводного интерфейса «nRF24L01»;
50. Документация на микросхему проводного интерфейса «ADM485EARZ»;
51. Дюльгер, Г.П. Вспомогательные репродуктивные технологии в воспроизводстве крупного рогатого скота / Г.П. Дюльгер, В.В. Храмцов, А.Г. Нежданов // Российский ветеринарный журнал. Сельскохозяйственные животные. – 2014. – №4, – С. 5–9.
52. Иванов Ю.Г. Радиотехническая система управления адресным обслуживанием животных. / Иванов Ю.Г. // Известия ТСХА. – 2005. – №1 – С. 67 – 69.
53. Карманный справочник инженера электронной техники. / К. Бриндли, Д. Карр. – М.: Додэка–XXI, 2007. – 480 с.
54. Киров А. В. Информационно−аналитическая система управления финансовой устойчивостью фирмы. // Финансовая аналитика. 2011. № 27. С. 11 − 14.
55. Кирсанов, В.В. Структурно−технологическое обоснование эффективного построения и функционирования доильного оборудования.: Автореф дис. ... д−ра тех. наук. / В.В. Кирсанов − М, 2001. − 47 с.
56. Комментарий к расчетам окупаемости реализации программы развития пилотных семейных молочных животноводческих ферм на базе крестьянских (фермерских) хозяйств. / Интернет–портал Министерства сельского хозяйства Российской Федерации / http://www.mcx.ru/documents/document/show_print/9144.260.htm.
57. Компьютерное проектирование электротехнических систем. / Л. П. Шичков, А. В. Дубровин, О. П. Мохова – М.: РГАЗУ, 2005. – 72 с.
58. Копопельцев, И. Г. Гистерограф − прибор для контроля за сократимостью матки у коров [Текст] / И. Г. Копопельцев, В. Н. Шулятьев // Ветеринария. − 2010. − № 5. − С. 42−44
59. Краснов, И.Н. Доильные аппараты. / − Ростов. Ростовский университет, 1974. – 162 с.

60. Краусп В. Р. Комплексная автоматизация в промышленном животноводстве. – М.: Машиностроение, 1980. – 214 с.

61. Кривенко, Д.В. Клинико–биохимические критерии оценки новорожденных телят после патологических родов / Д.В. Кривенко, Г.Г. Марченко, С.О. Лощинин // Вестник Саратовского госагроуниверситета им. Н.И. Вавилова. – 2012. – №3. – С. 19–21.

62. Кривенко, Д.В. Нарушение родового процесса у коров и его коррекция.: дис. доктор вет. наук: 16.00.07: / Кривенко Дмитрий Валентинович. – Краснодар, 2006 – С. 299.

63. Курзин, Н.Н. Механизм воздействия внешних электромагнитных полей на процессы стимулирования активности биологических объектов. / Н.Н. Курзин, Е.А. Рожков // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. − 2015. − № 108. − С. 702−724.

64. Курзин, Н.Н. Современные электрофизические методы и электротехнологические устройства для АПК Кубани. / Н.Н. Курзин, А.Э. Сулейманов, Д.А. Подповетная // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. − 2015. − № 112. − С. 1512−1541.

65. Курзин, Н.Н. Электрохимическая термодинамика клетки в условиях электромагнитного воздействия. / Н.Н. Курзин, Е.А. Рожков // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. − 2015. − № 108. − С. 1124−1146.

66. Логвинов, Д.Д. Беременность и роды у коров / Д. Д. Логвинов. – Киев: Урожай. – 1975. – С. 236.

67. Лощинин, С.О. Интранатальная асфиксия новорожденных телят в период патологических родов / С.О. Лощинин, В.Н. Чучин, В.С. Авдеенко, Д.В. Кривенко // Известия Горского государственного аграрного университета. – 2014. – №3 – Т.51. часть №3.– С. 150–156.

68. Лощинин, С.О. Клинико–биохимические индикаторы функционального состояния телят в неонатальный период после синдрома «трудные роды».: дис. канд. вет. наук: 06.02.06:/ Лощин Сергей Олегович. – Саратов, 2015 – С. 122.
69. Максимов Ю. Я. Физические величины и их единицы: Учебн.–справ. пособие − Авт.–сост. Чуваш. Ун–т. Чебоксары 1997. − 128 с.
70. Математические модели технических объектов. / В. А. Трудоношин, Н. В. Пивоварова. – М.: Высшая школа, 1986. – 158с.
71. Мельников С. В. Механизация и автоматизация животноводческих ферм. – Л.: Колос. Ленинг. отд–ние, 1978. – 560 с.
72. Методы классической и современной теории автоматического управления. Учебник в 3–х томах. Под редакцией Егунова Н.Д. – М.: Изд–во МГТУ им. Н.Э. Баумана, 2000. – 748 с.
73. Механизация и автоматизация животноводства (учебник). / А. Ф. Князев, Е. И. Резник, В. Ф. Сторчевой, С. В. Рыжов и др. − М.: Колос–С, 2004. – 375 с.
74. Механизация и электрификация животноводства. / Л. П. Карташов, А. И. Чугунов, А. А. Аверниев. – М.: Колос, 1997. – 368 с.
75. Молчанов, А.В. Критерии оценки физиологического состояния новорожденных телят после патологических родов / А.В. Молчанов, В.С. Авдеенко, С.О. Лощинин // Аграрный научный журнал – 2012. – №12 – С. 36–38.
76. Морозов Н. М. Стратегия развития механизации и автоматизации животноводства. // Техника и оборудование для села.– 2001. – №9. – С. 2 – 3.
77. Мусин А. М. Методические рекомендации по технико–экономической оценке автоматизированных технологических процессов животноводства. – М.: ВИЭСХ, 2003. – 44 с.
78. Мусин А. М. Технологический эффект автоматизации биотехнических систем производства. // Автоматизация сельскохозяйственного производства. Сб. докладов международной научно–технической конференции. – Углич, 2004. – С. 66 – 76.
79. Надежность электрорадиоизделий. / Справочник. ЦНИИ МО РФ. – М.: МО РФ, 2006. – 641 с.

80. Научные основы автоматизации животноводческих ферм. / Е. А. Воронин, В. Г. Вискуб, П. Н. Чинаев. – М.: МГАУ, – 2001. − 80 с.

81. Нежданов, А.Г. Влияние миотропных препаратов на сократительную функцию матки больных эндометритом коров. / А.Г. Нежданов, В.И. Михалев, В.В. Филин, Д.А. Ерин // Ветеринария. – 2013. – №1 – С. 33–35.

82. Нежданов, А.Г. Внутриутробная задержка развития эмбриона и плода у коров. / А.Г. Нежданов, В.И. Михалев, Н.Т. Климов, Е.В. Смирнова // Ветеринария. – 2014. – №3. – С. 36–39.

83. Нежданов, А.Г. Практическое руководство по обеспечению продуктивного здоровья крупного рогатого скота: учебное пособие / А.Г. Нежданов – Воронеж: «Антарес», 2011. – С. 220.

84. Нежданов, А.Г. Физиология и патология родов и послеродового периода у сельскохозяйственных животных. / А.Г. Нежданов – Воронеж, 1991. – С. 60.

85. Нефедов А.В. Интегральные микросхемы и их зарубежные аналоги. Справочник. Том 02. — Москва: ИП РадиоСофт. 2001 — 640 с.

86. Никитин, В.Я. К вопросу о профилактике и лечению акушерско–гинекологических заболеваний у коров. / В.Я. Никитин, В.Н. Белугин, Н.А. Писаренко, В.С. Скрипкин, Н.В. Федотов // Вестник АПК Ставрополья, – 2015. – №1 – С. 19–22.

87. Никольский В. В. Электродинамика и распространение радиоволн. – М.: Наука, 1978. – 543 с.

88. Нормы технологического проектирования предприятий крупного рогатого скота: НТП 1–99. МСХ РФ. – М.: 1999.− 152 с.

89. Основы автоматики: учебник для студентов средних специальных учебных заведений по специальности 3107 "Электрификация и автоматизация сельского хозяйства". / В. И. Загинайлов, Л. Н. Шеповалова; Под ред. Н. К. Петрова. – М.: Колос, 2001. – 198 с.

90. Павлов, В.А. Физиология воспроизводства крупного рогатого скота. / В.А. Павлов – М.: Россельхозиздат, 1984. – С. 208.

91. Пат. 134782 Российская Федерация, МПК A61D 17/00. Система определения на начала отела / Иванов Ю.Г., Голубятников В.А., Дюльгер Г.П., Сидоренко М.С.; заявитель и патентообладатель ФГБОУ ВПО РГАУ – МСХА имени К.А. Тимирязева. – № 2013125623/13 заявл. 04.06.2013; опубл. 27.11.2013, Бюл. № 33

92. Пат. US 2010/0331739 A1 США. Method and system for predicting calving / E. Maltz.

93. Пирогова Е. В. Проектирование и технология печатных плат. – М.: ФОРУМ – ИНФРА–М, 2005. – 524 с.

94. Планирование эксперимента в исследованиях сельскохозяйственных процессов. / С. В. Мельников, В. Р. Алешин, П. М. Рощин. − Л.: Колос, 1980. – 166 с.

95. Планирование эксперимента в условиях неоднородностей. / Е. В. Марков, А. Н. Лисенков. − М.: Наука, 1973. – 40 с.

96. Планирование эксперимента при поиске оптимальных условий / Ю. П. Адлер, К. В. Маркова, Ю. В. Грановский – М.: Наука, 1971. – 156 с.

97. Попов А. А. Прогрессивным технологиями содержания коров – широкое внедрение // Техника в сельском хозяйстве. – 1986. №3. С. 16.

98. Постановление Правительства РФ от 12.10.2004 № 539 «О порядке регистрации радиоэлектронных средств и высокочастотных устройств» (в ред. Пост. Правительства РФ от 25.07.2007 № 476, от 13.10.2011 № 837).

99. Правила устройства электроустановок. 7 издание – М.: Изд−во НЦ ЭНАС, 2004. – 600 с.

100. Практикум по акушерству, гинекологии и биотехнике репродукции животных: учебное пособие вузов / В.Я. Никитин – Москва: Изд–во РГАУ–МСХА имени К.А. Тимирязева, 2014. – С. 331.

101. Радиотехнические цепи и сигналы / Д. В. Васильев, Н. Р. Витоль, А. П. Горошенков и др.; Под ред. К. А. Самойло. – М.: Радио и связь, 1982. – 240 с.

102. Родионов Г. В. Справочник по молочному скотоводству. – М.: Агроконсалт, 2001. – 200 с.

103. Сапаров В. Е. Дипломный проект от А до Я – М.: СОЛОН–Пресс, 2004, – 224 с.

104. Свидетельство о государственной регистрации программы для ЭВМ № 2015616713 от 19 июня 2015 г. Драйвер микросхемы радиоинтерфейса nRF24L01 для микроконтроллеров с ядром AVR / Иванов Ю.Г., Дюльгер Г.П., Голубятников В.А., Сидоренко М.С.

105. Севастьянов Б. А. Курс теории вероятностей и математической статистики. – М.: Наука, 1982. – 146 с.

106. Семиволос, А.М. Радиоэлектронные способы коррекции сократительной функции матки и диагностики беременности у коров.: дис. ... д−ра вет. наук: 16.00.07 / Семиволос Александр Мефодьевич. − Саратов, 1999. − 415 с.

107. Сергованцев В. Т. Компьютеризация сельскохозяйственного производства. – М.: Колос, 2005. – 310 с.

108. Сидоренко М.С. Дистанционный контроль сигналов коров / Ю.Г. Иванов, В.А. Голубятников, М.С. Сидоренко // Сельский механизатор – 2015. – № 10. – С. 26 − 27.

109. Сидоренко М.С. Дистанционный мониторинг функциональной деятельности органов промежности коровы / Ю.Г. Иванов, В.А. Голубятников, М.С. Сидоренко // Механизация и электрификация сельского хозяйства – 2014. – № 4. – С. 16 – 19.

110. Сидоренко М.С. Исследование микропроцессорной системы контроля функциональной деятельности органов промежности коровы / Ю.Г. Иванов, Г.П. Дюльгер, М.С. Сидоренко // Техника в сельском хозяйстве – 2014. – № 6. – С. 11 – 13.

111. Сидоренко М.С. Исследования микропроцессорной системы дистанционного мониторинга сигналов коров / Ю.Г. Иванов, В.А. Голубятников, М.С. Сидоренко // Вестник ФГБОУ ВПО МГАУ имени В.П. Горячкина – 2015. – № 2 (66). – С. 7 – 14.

112. Сидоренко М.С. Микропроцессорная система контроля функциональной деятельности органов промежности коровы / Ю.Г. Иванов, Г.П. Дюльгер, В.А. Голубятников, М.С. Сидоренко // Вестник ВИЭСХ – 2014. – № 3 (16). – С. 60 – 63.
113. Сидоренко М.С. Первичные электронно–технические средства для определения родового акта коровы / Ю.Г. Иванов, В.А. Голубятников, М.С. Сидоренко // Материалы Всероссийской конференции студентов, аспирантов и ученых с международным участием 13 декабря 2012г. «Основные направления развития техники и технологии в АПК, легкой и пищевой промышленности», г. Княгинино, НГИЭИ – 2013. – С. 188 – 193.
114. Сидоренко М.С. Радиотехнические системы мониторинга животных на молочной ферме / Ю.Г. Иванов, Г.П. Дюльгер, М.С. Сидоренко // Материалы Международной научно–практической конференции, посвященной 85–летию академика К.У. Медеубекова «Зоотехническая наука Казахстана: прошлое, настоящее, будущее», г. Алматы – 2014. – С. 295 – 300.
115. Сидоренко М.С. Системы дистанционного контроля сигналов коров / Ю.Г. Иванов, Г.П. Дюльгер, М.С. Сидоренко // Зоотехния – 2014. – № 12. – С. 6 – 7.
116. Сидоренко М.С. Структура технических и программных средств дистанционного контроля сигналов коров / Ю.Г. Иванов, В.А. Голубятников, М.С. Сидоренко // Вестник ВНИИМЖ – 2015. – № 3 (19). – С. 109 − 112.
117. Симарев Ю. Ипользование техники при поточно–цеховой системе содержания коров // Молочное и мясное скотоводство. – 1984. №.9. С. 10 – 13.
118. Славин Р. М. Научные основы автоматизации производства в животноводстве и птицеводстве. – М.: Колос, 1974. − 226 с.
119. Справочник по математике. / Г. Корн, Т. Корн. – М.: Наука, 1974. – 708 с.
120. Стребков, Д.С. Научно−технические проблемы автоматизации стационарных технологических процессов в сельском хозяйстве. / Д.С. Стребков // Автоматизация сельскохозяйственного производства. Материалы докладов международной научно технической конференции, т.1. – М.: ВИМ, 1997.−С. 79−97.

121. Теоретическая механика / М. И. Белов, Б. В. Пылаев. – М.: Издательство РГАУ–МСХА. 2011. – 296 с.

122. Терентьева, Н.Ю. Профилактическая эффективность фитопрепаратов при патологии послеродового периода у высокопродуктивных молочных коров.: дис. ... к.вет.н. наук: 16.00.07 / Терентьева, Н.Ю. − Ульяновск, 2004. − 173 с.

123. Технология и механизация молочного животноводства. / Е. Е. Хазанов, В. В. Гордеев, В. Е. Хазанов. – СПб.: Лань, 2010. – 352 с.

124. Угрюмов Е. П. Цифровая схемотехника – Санкт–Петербург: БВХ–Петербург, 2002. – 518 с.

125. Ужик, В.Ф. Механизация и автоматизация в животноводстве. / А.Н. Макаренко, В.Ф. Ужик, О.А. Чехунов, Ю.В. Саенко, Е.А. Мартынов, А.И. Скляров, К.Н. Путиенко // Белгород, − 2012.

126. Ужик, В.Ф. Механизация и автоматизация в животноводстве. / А.Н. Макаренко, В.Ф. Ужик, О.А. Чехунов, Ю.В. Саенко, Е.А. Мартынов, А.И. Скляров, К.Н. Путиенко, // Практикум для подготовки бакалавров по направлению 111100.62 − Зоотехния / Белгород, − 2012.

127. Ужик, О.В. Автоматизация элементов технологии и экономическая эффективность производства молока. / О.В. Ужик, Я.В. Ужик // В сборнике: Проблемы и перспективы инновационного развития агроинженерии, энергоэффективности и IT−технологий. Материалы XVIII Международной научно−производственной конференции. − 2014. − С. 203.

128. Ужик, О.В. К оценке эффективности использования технических средств в молочном скотоводстве. / А.И. Завражнов, О.В. Ужик // Вестник Мичуринского государственного аграрного университета. − 2013. − № 3. − С. 51−56.

129. Ужик, О.В. Механизация и автоматизация животноводства. / А.Н. Макаренко, В.Ф. Ужик, А.И. Скляров, О.В. Ужик, В.И. Борозенцев, О.А. Чехунов, Ю.В. Саенко, Е.А. Мартынов, К.Н. Путиенко // Белгород, − 2015.

130. Ужик, О.В. Механизация и автоматизация технологических процессов растениеводства и животноводства. / А.Н. Макаренко, В.Ф. Ужик, О.А. Чехунов,

О.А. Скляров, Ю.В. Саенко, Е.А. Мартынов, О.В. Ужик, К.Н. Путиенко // Белгород, − 2014.

131. Ужик, О.В. Разработка и теоретическое исследование технологий и технических средств функционирования молочного скотоводства. / О.В. Ужик // Автореферат дис. ... доктора технических наук: 05.20.01. Мичуринский государственный аграрный университет. Мичуринск, − 2012

132. Ужик, О.В. Технико−технологическое обеспечение молочного скотоводства. / О.В. Ужик // Вестник Всероссийского научно−исследовательского института механизации животноводства. − 2013. − № 2 (10). − С. 195−204.

133. Ужик, О.В. Технические средства для молочного скотоводства. / О.В. Ужик // В сборнике: Инновационные технологии и технические средства для АПК. материалы Международной научно−практической конференции молодых ученых и специалистов. Воронежский государственный аграрный университет, Совет молодых ученых и специалистов. − 2014. − С. 3−11.

134. Учеваткин, А.И. Автоматизированные энергосберегающие технологии и система электрооборудования линий первичной обработки молока на фермах.: Автореф дис. ... д−ра тех. наук. / А.И. Учеваткин − М, 1998. − 44 с.

135. Федеральный закон «О связи». Статья 24. Выделение полос радиочастот и присвоение (назначение) радиочастот или радиочастотных каналов.

136. Финансовый анализ. / А. Ф. Ионова, Н. Н. Селезнева. – М.: Проспект, 2009. − 623 с.

137. Финансовый анализ. / В. Р. Банк, С. В. Банк, Л. В. Тараскина. – М.: Проспект, 2009. – 343 с.

138. Финансовый анализ. / Л. С. Васильева, М. В. Петровская.– М.: КНОРУС, 2008. − 804 с.

139. Цой Ю. А. Задачи исследования по механизации и автоматизации молочного животноводства // Техника в сельском хозяйстве. 1988. №5. С. 25 – 26.

140. Шичков Л. П. Основы информатики и информационных технологий. – М.: РГАЗУ, 2002. – 78 с.

141. Электрооборудование и средства автоматизации сельскохозяйственной техники. / Л. П. Шичков, А. П. Коломиец – М.: Колос, 1995. – 368 с.
142. A practical way to detect approaching calving of the dairy cow by a behaviour sensor. / Maltz E., Antler A. // Proc. Precision livestock farming ’07, Edited by: S. Cox, Academic Publishers. − 2007. P. 141 − 146.
143. Armstong, D. Automatic apply of teat cups. / D. Armstong, R. Middel, R. Oenema // PCT Patent WO 85/02973, AO 1j7/00, − 1985.
144. Automated monitoring of dairy cows’ behavioral patterns as a tool for management. / Livshin N., Aizinbud E., Tinsky M., Bargai U., Maltz E. // Proceedings, itafe '03, International Congress on Information Technology in Agriculture, Food and Environment, 7 to 10 October 2003, Izmir, Turkey.
145. Automation of oestrous detection in dairy cows: a review. / Frik R., Stamer E., Junge W., Kreieter J. // Livestock Production Science, − 2002, P. 219 − 232.
146. Calving Emergencies in Beef Cattle: Identification and Prevention / Whittier W. D., Currin N. M., Currin J. F., Hall J. B. – Virginia State University, 2009.
147. Current and Near Term Technologies for Automated Recording of Animal Data for Precision Dairy Farming. / Katz G., Arazi A., Pinski N., Halachmi I., Schmilovitz Z., Aizinbud E., Maltz E. // ADSA, San Antonio, TX, USA, − 2007. J. Dairy Sci. Vol. 90, Suppl. 1, P. 377.
148. Lying behaviour of dairy cows under different housing systems and physiological conditions. / Livshin N., Antler A., Zion B., Stojanovski G., Bunevski G., E. Maltz. // Proc. 2nd ECPLF, 9−12 June 2005 in Uppsala, Sweden, P. 305 − 311.
149. Maltz E. Novel Technologies: Sensors, Data and Precision Dairy Farming / Maltz E. // The First North American Conference on Precision Dairy Management 2010
150. Review: Cost of reproductive diseases and conditions in cattle / Bellows D. S., Ott S. L., Bellows R. A. // The Professional Animal Scientist. − 2002. − Vol. 18. – P. 26 – 32.
151. Safer oestrus detection with sensor aided ALT−pedometer. / Breheme E., Stollberg U., Holtz R., Scleusener T. // Proceedings of Third International Workshop on Smart Sensors in Livestock Monitoring − 2004. Leuven, Belgium. P. 43 − 46.

152. Walter G. Jung OP AMP APPLICATIONS – Analog Devices, 2002.

ПРИЛОЖЕНИЯ

Приложение А: Документы об использовании результатов исследований.

Проректор по научной работе
ФГБОУ ВО РГАУ-МСХА
имени К.А. Тимирязева
_____ А.В. Голубев
«08» 09 _____ 2015 г.

Справка
об апробации и использовании результатов исследований по кандидатской диссертации, выполненной аспирантом кафедры «Автоматизация и механизация животноводства» РГАУ-МСХА имени К.А. Тимирязева Сидоренко М.С. по теме: «Разработка электротехнических средств для определения времени начала родов у коров».

В РГАУ-МСХА имени К.А. Тимирязева в условиях молочной фермы зоостанции проведена производственная проверка электротехнических средств для определения времени начала родов у коров с использованием: датчика мышечного напряжения хвоста животного, приемо-передающего устройства и ПК. Проведенные в период май- август 2015 года исследования и производственная проверка показали работоспособность электротехнических средств и регистрацию ими времени начала родов коров.

Результаты исследований представляют научный и практический интерес и могут быть использованы для производственного применения. Получен патент РФ и свидетельство о государственной регистрации программы для ЭВМ.

Материалы диссертации используются на кафедре «Автоматизация и механизация животноводства» в научных исследованиях, а так же при чтении лекций и выполнении курсовых и дипломных проектов студентами факультета зоотехнии и биологии.

Декан факультета зоотехнии и биологии, д.с.-х.н., профессор — Ю.А. Юлдашбаев

Профессор кафедры автоматизации и механизации животноводства, д.т.н. — В.А. Воробьев

Руководитель зоостанции — С.Н. Варлыгин

Ведущий инженер кафедры — Е.В. Машошина

«Утверждаю»
Проректор по научной работе
ФГБОУ ВПО ЧГСХА
_____В.Г. Семенов
«23» сентября 2015 г.

СПРАВКА
об использовании результатов исследований
по диссертационной работе Сидоренко М.С.

Справка дана в том, что результаты исследований научно-исследовательской работы по разработке электротехнических средств для определения времени начала родов у коров, алгоритм функционирования устройства и математические модели, выполненные в виде кандидатской диссертации Сидоренко М.С. – аспирантом кафедры автоматизации и механизации животноводства ФГБОУ ВО РГАУ–МСХА имени К.А.Тимирязева, используются в учебном процессе кафедры механизация, электрификации и автоматизации сельскохозяйственного производства ФГБОУ ВПО ЧГСХА при чтении лекций, выполнении курсовых и дипломных проектов.

Заведующий кафедрой МЭАСХП
к.т.н., доцент С.Н. Мардарьев

«УТВЕРЖДАЮ»
Ректор ГОУ ВО «Нижегородский государственный инженерно-экономический университет»
д.э.н., профессор А.Е. Шамин

«21» 09 2015 г.

Акт

Комиссия, в составе заведующего кафедрой электрификации и автоматизации к.т.н., доцента В. Л. Осокина, д.т.н., профессора А. С. Серебрякова и старшего преподавателя Т. В. Шиловой, составила настоящий акт о том, что результаты научно-исследовательской работы по разработке электротехнических средств для определения времени начала родов у коров, выполненной в виде кандидатской диссертации Сидоренко М. С., аспирантом кафедры автоматизации и механизации животноводства ФГБОУ РГАУ МСХА имени К. А. Тимирязева, используются в учебном процессе при чтении лекций, выполнении курсовых и дипломных проектов.

Зав. кафедрой электрификации и автоматизации, к.т.н., доцент — В. Л. Осокин

д.т.н., профессор — А. С. Серебряков

ст. преподаватель — Т. В. Шилова

«Утверждаю»

Проректор по учебной работе

ФГБОУ ВО «Великолукская ГСХА»

_____ Павлов А.Н.

_____09_____ 2015 г.

СПРАВКА

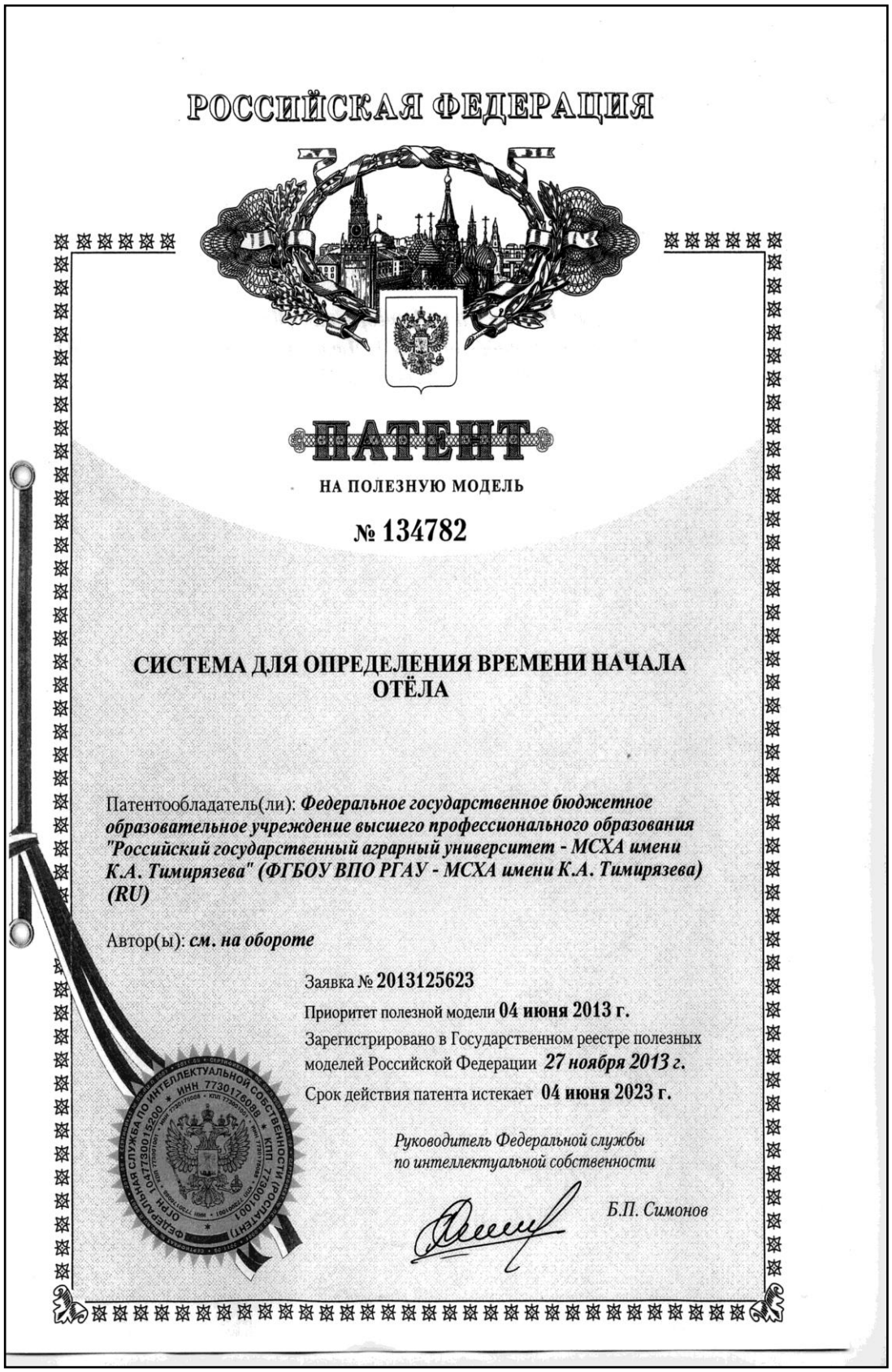
об использовании результатов исследований

по диссертационной работе Сидоренко М.С.

Справка дана в том, что результаты научно-исследовательской работы Сидоренко М.С., аспиранта кафедры автоматизации и механизации животноводства ФГБОУ ВО РГАУ–МСХА имени К.А. Тимирязева, по разработке электротехнических средств для определения времени начала родов у коров, в т.ч. результаты теоретических и экспериментальных исследований, выполненные в виде кандидатской диссертации используются в учебном процессе кафедры «Механизация животноводства и применение электрической энергии в сельском хозяйстве» ФГБОУ ВО «Великолукская ГСХА» при чтении лекций и выполнении курсовых проектов.

Зав. кафедрой «Механизация животноводства
и применение электроэнергии в сельском
хозяйстве» ФГБОУ ВО «Великолукская ГСХА»,
доктор технических наук, доцент Самарин Г.Н.

Приложение Б: Патенты и свидетельства.



РОССИЙСКАЯ ФЕДЕРАЦИЯ

ПАТЕНТ

НА ПОЛЕЗНУЮ МОДЕЛЬ

№ 134782

СИСТЕМА ДЛЯ ОПРЕДЕЛЕНИЯ ВРЕМЕНИ НАЧАЛА ОТЁЛА

Патентообладатель(ли): *Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования "Российский государственный аграрный университет - МСХА имени К.А. Тимирязева" (ФГБОУ ВПО РГАУ - МСХА имени К.А. Тимирязева) (RU)*

Автор(ы): *см. на обороте*

Заявка № **2013125623**

Приоритет полезной модели **04 июня 2013 г.**

Зарегистрировано в Государственном реестре полезных моделей Российской Федерации ***27 ноября 2013 г.***

Срок действия патента истекает **04 июня 2023 г.**

Руководитель Федеральной службы по интеллектуальной собственности

Б.П. Симонов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2015616713

Драйвер микросхемы радиоинтерфейса nRF24L01 для микроконтроллеров с ядром AVR

Правообладатель: ***Федеральное государственное бюджетное образовательное учреждение высшего образования «Российский государственный аграрный университет - МСХА имени К.А. Тимирязева» (ФГБОУ ВО РГАУ-МСХА имени К.А. Тимирязева) (RU)***

Авторы: ***Иванов Юрий Григорьевич (RU), Сидоренко Михаил Сергеевич (RU), Голубятников Вадим Александрович (RU), Дюльгер Георгий Петрович (RU)***

Заявка № **2015612502**

Дата поступления **01 апреля 2015 г.**

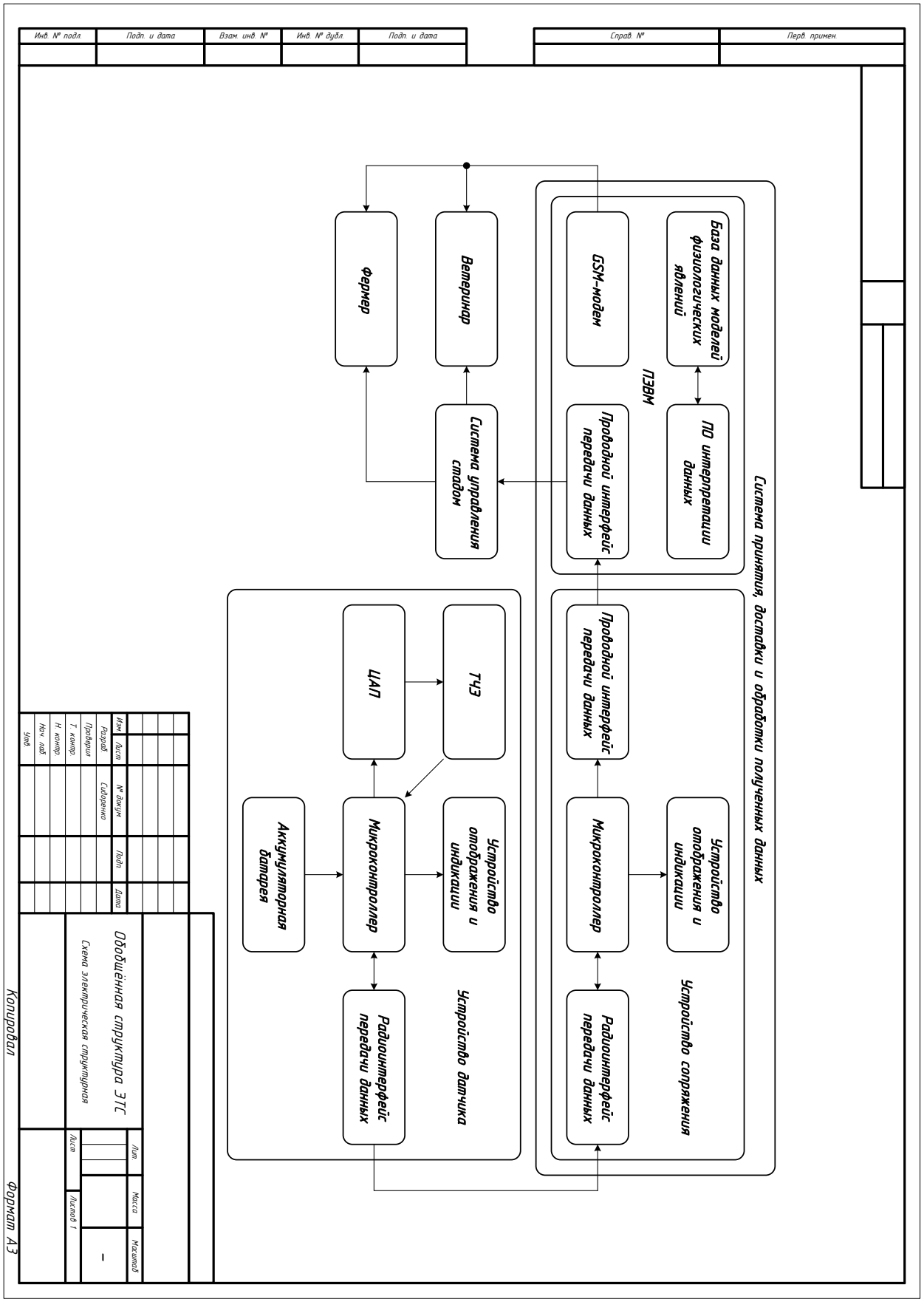
Дата государственной регистрации

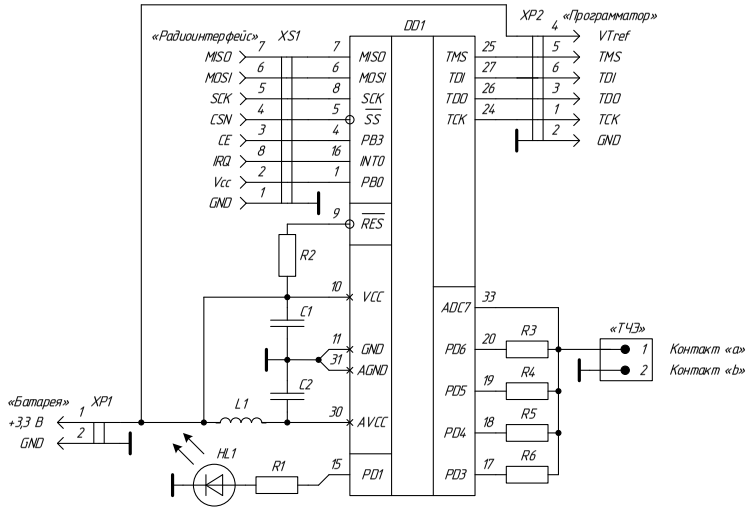
в Реестре программ для ЭВМ ***19 июня 2015 г.***

Врио руководителя Федеральной службы по интеллектуальной собственности

Л.Л. Кирий

Приложение В: Структурные и принципиальные схемы.





Поз. Обозн.	Наименование	Кол.	Примечание
C1, C2	Конденсаторы GCM155R71C104KA55D "Murata"	2	0402, 0,1 мкФ
DD1	Микросхема ATmega16L-8MU "Atmel"	1	QFN44
L1	Индуктивность CM322522-100KL "Bourns"	1	1210, 10 мкГ
	Резисторы		
R1	RC0402JR-0722R "Yageo"	1	0402, 22 Ом
R2	RG1005P-4,73-B-T5 "Susumu"	1	0402, 4,7 кОм
R3	RG1005P-103-B-T5 "Susumu"	1	0402, 10 кОм
R4	RG1005P-223-B-T5 "Susumu"	1	0402, 22 кОм
R5	RG1005P-393-B-T5 "Susumu"	1	0402, 39 кОм
R6	RG1005P-823-B-T5 "Susumu"	1	0402, 82 кОм
HL1	Светодиод KPHHS-1005CGCK "Kingbright"	1	0402, Зелёный
	Вилки		
XP1	M50-3600542 "Harwin"	1	Обрезать до PLLD127-6S
XP2	M50-3600542 "Harwin"	1	Обрезать до PLLD127-2S
XS1	Розетка PBD16 "Connfly Electronic"	1	DS1023-2x8

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Сидоренко		
Проверил				
Т. контр.				
Н. контр.				
Нач. лаб.				
Утв.				

Устройство датчика

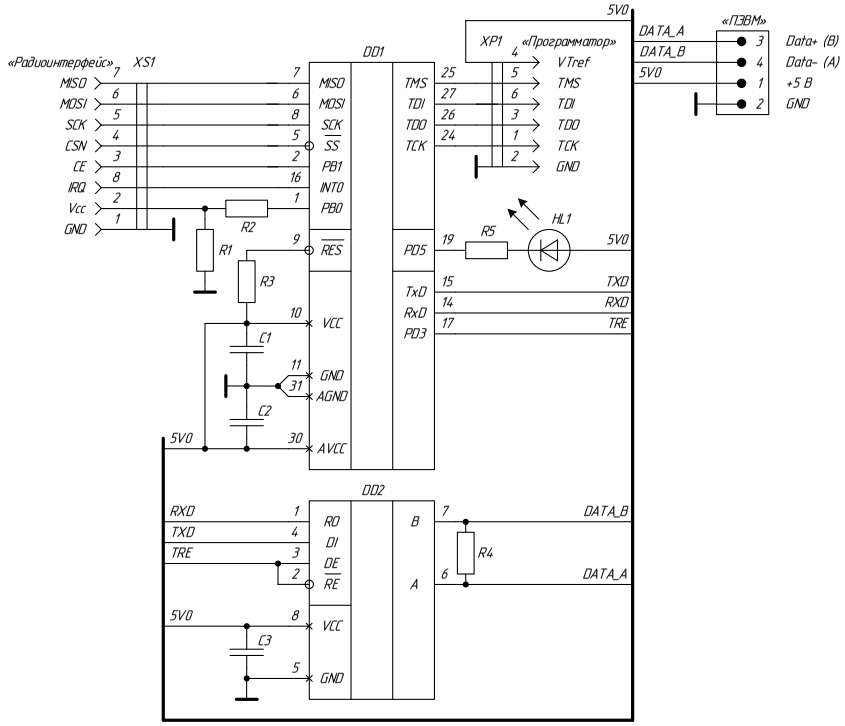
Схема электрическая принципиальная

Лит.	Масса	Масштаб
		–
Лист	Листов 1	

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал

Формат А3



Поз. Обозн.	Наименование	Кол.	Примечание
C1 – C3	Конденсаторы GCM155R71C104KA55D "Murata"	3	0402, 0,1 мкФ
	Микросхемы		
DD1	Микросхема ATmega16-16MU "Atmel"	1	QFN44
DD2	Микросхема ADM485EARZ "Analog Devices"	1	SOIC8
	Резисторы		
R1	RG1005P-100-B-T5 "Susumu"	1	0402, 100 Ом
R2	RG1005P-50-B-T5 "Susumu"	1	0402, 50 Ом
R3	RG1005P-4,73-B-T5 "Susumu"	1	0402, 4,7 кОм
R4	RG1005P-120-B-T5 "Susumu"	1	0402, 120 Ом
R5	RC0402JR-0722R "Yageo"	1	0402, 22 Ом
HL1	Светодиод KPHHS-1005CGCK "Kingbright"	1	0402, Зелёный
XP1	M50-3600542 "Harwin"	1	Обрезать до PLLD127-6S
XS1	Розетка PBD16 "Connfly Electronic"	1	DS1023-2x8

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Сидоренко		
Проверил				
Т. контр.				
Н. контр.				
Нач. лаб.				
Утв.				

Устройство сопряжения

Схема электрическая принципиальная

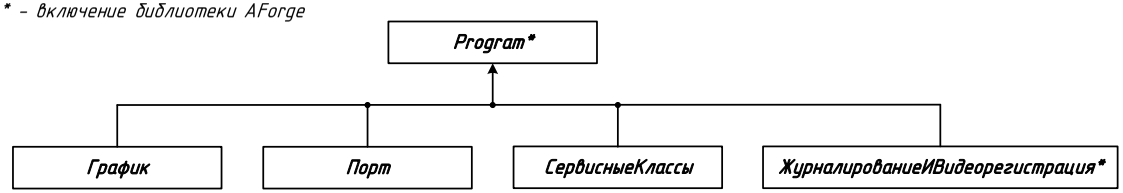
Лит.	Масса	Масштаб
		–
Лист	Листов 1	

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал

Формат А3

Примечание: * - включение библиотеки AForge



Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Сидоренко		
Проверил				
Т. контр.				
Н. контр.				
Нач. лаб.				
Утв.				

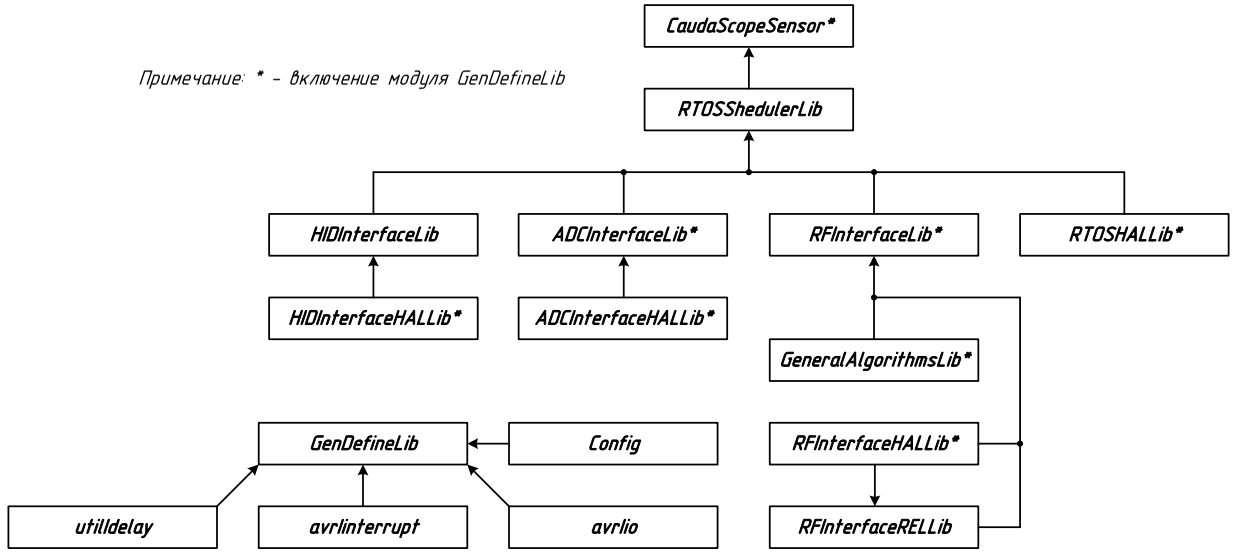
Структура специального программного обеспечения ПЭВМ

Лит.	Масса	Масштаб
		–
Лист	Листов 1	

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал

Формат А3



Примечание: * - включение модуля GenDefineLib

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Сидоренко		
Проверил				
Т. контр.				
Н. контр.				
Нач. лаб.				
Утв.				

Структура микропрограммы устройства датчика

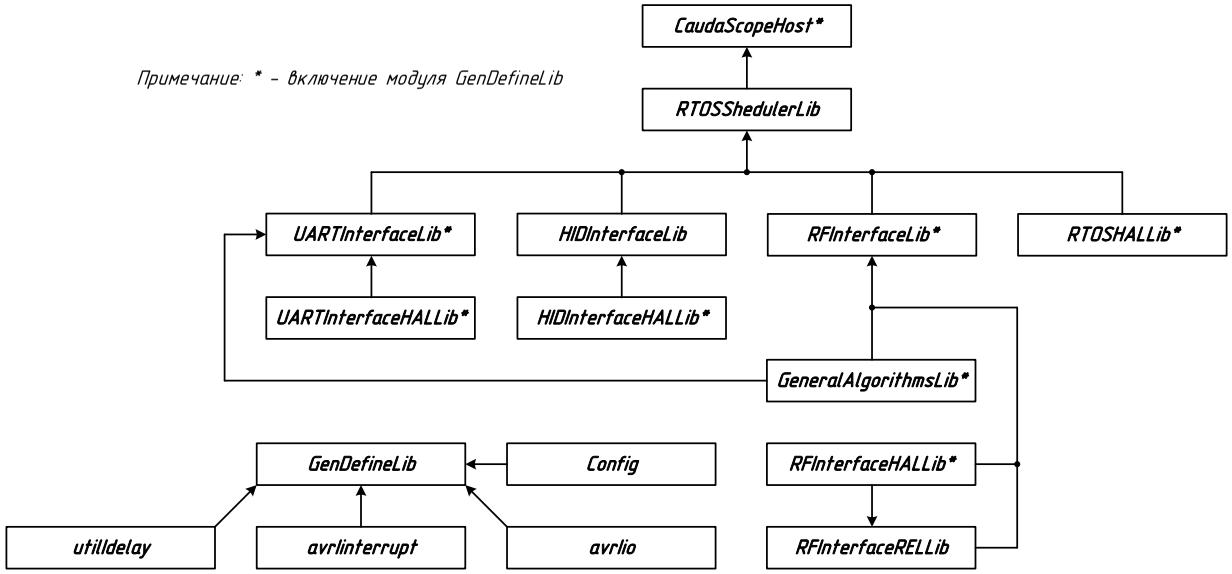
Лит.	Масса	Масштаб
		–
Лист	Листов 1	

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал

Формат А3

Примечание: * - включение модуля GenDefineLib



Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Сидоренко		
Проверил				
Т. контр.				
Н. контр.				
Нач. лаб.				
Утв.				

Структура микропрограммы устройства сопряжения

Лит.	Масса	Масштаб
		–
Лист	Листов 1	

Перв. примен. | Справ. № | Подп. и дата | Инв. № дубл. | Взам. инв. № | Подп. и дата | Инв. № подл.

Копировал

Формат А3

Приложение Г: Листинги.

ЛИСТИНГ МИКРОПРОГРАММЫ ЭТИП

Листинг файла ADCInterfaceHALLib.c

```
/*
 * Назначение: Файл модуля абстракции АЦП датчика.
 * Имя файла: ADCInterfaceHALLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 30.04.2015 г.
 * Список изменений:
 * 30.04.2015 г. − Исходная версия.
*/

#include "ADCInterfaceHALLib.h"

/*****************************************************************************
************
 * Прототип функции: void ADCIHInit(void);
 * Назначение: Функция инициализации АЦП.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция инициализации АЦП.

*****************************************************************************
***********/
void ADCIHInit(void)
{
	CLEAR_BITS(SFIOR,(1<<ADTS2)|(1<<ADTS1)|(1<<ADTS0));
	MODIFY_REG(ADMUX,((1<<REFS1)|(1<<ADLAR)|(1<<MUX4)|(1<<MUX3)),((1<<REFS
0)|(1<<MUX2)|(1<<MUX1)|(1<<MUX0)));
	MODIFY_REG(ADCSRA,((1<<ADSC)|(1<<ADATE)),((1<<ADEN)|(1<<ADIE)|(1<<ADPS2
)|(1<<ADPS1)|(1<<ADPS0)));
	SET_BIT(ACSR,ACD);
}

/*****************************************************************************
************
 * Прототип функции: uint16_t ADCIHHandler(void);
 * Назначение: Функция обработки прерывания АЦП.
 * Аргументы: Нет.
 * Возвращаемые значения: Оцифрованное значение.
 * Описание: Функция обработки прерывания АЦП.

*****************************************************************************
***********/
uint16_t ADCIHHandler(void)
{
	register uint8_t LowByte=ADCL;
	register uint8_t HighByte=ADCH;
	return (uint16_t)((HighByte<<8)|LowByte);
}
```

Листинг заголовочного файла ADCInterfaceHALLib.h

```
/*
 * Назначение: Заголовочный файл модуля абстракции АЦП датчика.
 * Имя файла: ADCInterfaceHALLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 30.04.2015 г.
 * Список изменений:
 * 30.04.2015 г. − Исходная версия.
 */

#ifndef ADCINTERFACEHALLIB_H_
#define ADCINTERFACEHALLIB_H_

#include "GenDefineLib.h"

#define ADCIHResistValDDR_0_00()
	CLEAR_BITS(DDRD,((1<<PD3)|(1<<PD4)|(1<<PD5)|(1<<PD6)))		/*Установка
порта DDR матрицы резисторов сопротивлением 0 кОм.*/
#define ADCIHResistValPORTD_0_00()
	CLEAR_BITS(PORTD,((1<<PD3)|(1<<PD4)|(1<<PD5)|(1<<PD6)))		/*Установка
порта PORTD матрицы резисторов сопротивлением 0 кОм.*/
#define ADCIHResistValDDR_82_00()
	MODIFY_REG(DDRD,((1<<PD3)|(1<<PD4)|(1<<PD5)),((1<<PD6)))	/*Установка порта
DDR матрицы резисторов сопротивлением 82,00 кОм.*/
#define ADCIHResistValPORTD_82_00()
	MODIFY_REG(PORTD,((1<<PD3)|(1<<PD4)|(1<<PD5)),((1<<PD6)))	/*Установка порта
PORTD матрицы резисторов сопротивлением 82,00 кОм.*/
#define ADCIHResistValDDR_39_00()
	MODIFY_REG(DDRD,((1<<PD3)|(1<<PD4)|(1<<PD6)),(1<<PD5))		/*Установка
порта DDR матрицы резисторов сопротивлением 39,00 кОм.*/
#define ADCIHResistValPORTD_39_00()
	MODIFY_REG(PORTD,((1<<PD3)|(1<<PD4)|(1<<PD6)),(1<<PD5))		/*Установка
порта PORTD матрицы резисторов сопротивлением 39,00 кОм.*/
#define ADCIHResistValDDR_26_43()
	MODIFY_REG(DDRD,((1<<PD3)|(1<<PD4)),((1<<PD5)|(1<<PD6)))	/*Установка порта
DDR матрицы резисторов сопротивлением 26,43 кОм.*/
#define ADCIHResistValPORTD_26_43()
	MODIFY_REG(PORTD,((1<<PD3)|(1<<PD4)),((1<<PD5)|(1<<PD6)))	/*Установка порта
PORTD матрицы резисторов сопротивлением 26,43 кОм.*/
#define ADCIHResistValDDR_22_00()
	MODIFY_REG(DDRD,((1<<PD3)|(1<<PD5)|(1<<PD6)),((1<<PD4)))	/*Установка порта
DDR матрицы резисторов сопротивлением 22,00 кОм.*/
#define ADCIHResistValPORTD_22_00()
	MODIFY_REG(PORTD,((1<<PD3)|(1<<PD5)|(1<<PD6)),((1<<PD4)))	/*Установка порта
PORTD матрицы резисторов сопротивлением 22,00 кОм.*/
#define ADCIHResistValDDR_17_35()
	MODIFY_REG(DDRD,((1<<PD3)|(1<<PD5)),((1<<PD4)|(1<<PD6)))	/*Установка порта
DDR матрицы резисторов сопротивлением 17,35 кОм.*/
#define ADCIHResistValPORTD_17_35()
	MODIFY_REG(PORTD,((1<<PD3)|(1<<PD5)),((1<<PD4)|(1<<PD6)))	/*Установка порта
PORTD матрицы резисторов сопротивлением 17,35 кОм.*/
```

```
#define ADCIHResistValDDR_14_07()
        MODIFY_REG(DDRD,((1<<PD3)|(1<<PD6)),((1<<PD4)|(1<<PD5)))    /*Установка порта
DDR матрицы резисторов сопротивлением 14,07 кОм.*/
#define ADCIHResistValPORTD_14_07()
        MODIFY_REG(PORTD,((1<<PD3)|(1<<PD6)),((1<<PD4)|(1<<PD5)))   /*Установка порта
PORTD матрицы резисторов сопротивлением 14,07 кОм.*/
#define ADCIHResistValDDR_12_01()
        MODIFY_REG(DDRD,((1<<PD3)),((1<<PD4)|(1<<PD5)|(1<<PD6)))    /*Установка порта
DDR матрицы резисторов сопротивлением 12,01 кОм.*/
#define ADCIHResistValPORTD_12_01()
        MODIFY_REG(PORTD,((1<<PD3)),((1<<PD4)|(1<<PD5)|(1<<PD6)))   /*Установка порта
PORTD матрицы резисторов сопротивлением 12,01 кОм.*/
#define ADCIHResistValDDR_10_00()
        MODIFY_REG(DDRD,((1<<PD4)|(1<<PD5)|(1<<PD6)),((1<<PD3)))    /*Установка порта
DDR матрицы резисторов сопротивлением 10,00 кОм.*/
#define ADCIHResistValPORTD_10_00()
        MODIFY_REG(PORTD,((1<<PD4)|(1<<PD5)|(1<<PD6)),((1<<PD3)))   /*Установка порта
PORTD матрицы резисторов сопротивлением 10,00 кОм.*/
#define ADCIHResistValDDR_8_91()
        MODIFY_REG(DDRD,((1<<PD4)|(1<<PD5)),((1<<PD3)|(1<<PD6)))    /*Установка порта
DDR матрицы резисторов сопротивлением 8,91 кОм.*/
#define ADCIHResistValPORTD_8_91()
        MODIFY_REG(PORTD,((1<<PD4)|(1<<PD5)),((1<<PD3)|(1<<PD6)))   /*Установка порта
PORTD матрицы резисторов сопротивлением 8,91 кОм.*/
#define ADCIHResistValDDR_7_96()
        MODIFY_REG(DDRD,((1<<PD4)|(1<<PD6)),((1<<PD3)|(1<<PD5)))    /*Установка порта
DDR матрицы резисторов сопротивлением 7,96 кОм.*/
#define ADCIHResistValPORTD_7_96()
        MODIFY_REG(PORTD,((1<<PD4)|(1<<PD6)),((1<<PD3)|(1<<PD5)))   /*Установка порта
PORTD матрицы резисторов сопротивлением 7,96 кОм.*/
#define ADCIHResistValDDR_7_25()
        MODIFY_REG(DDRD,((1<<PD4)),((1<<PD3)|(1<<PD5)|(1<<PD6)))    /*Установка порта
DDR матрицы резисторов сопротивлением 7,25 кОм.*/
#define ADCIHResistValPORTD_7_25()
        MODIFY_REG(PORTD,((1<<PD4)),((1<<PD3)|(1<<PD5)|(1<<PD6)))   /*Установка порта
PORTD матрицы резисторов сопротивлением 7,25 кОм.*/
#define ADCIHResistValDDR_6_88()
        MODIFY_REG(DDRD,((1<<PD5)|(1<<PD6)),((1<<PD3)|(1<<PD4)))    /*Установка порта
DDR матрицы резисторов сопротивлением 6,88 кОм.*/
#define ADCIHResistValPORTD_6_88()
        MODIFY_REG(PORTD,((1<<PD5)|(1<<PD6)),((1<<PD3)|(1<<PD4)))   /*Установка порта
PORTD матрицы резисторов сопротивлением 6,88 кОм.*/
#define ADCIHResistValDDR_6_34()
        MODIFY_REG(DDRD,((1<<PD5)),((1<<PD3)|(1<<PD4)|(1<<PD6)))    /*Установка порта
DDR матрицы резисторов сопротивлением 6,34 кОм.*/
#define ADCIHResistValPORTD_6_34()
        MODIFY_REG(PORTD,((1<<PD5)),((1<<PD3)|(1<<PD4)|(1<<PD6)))   /*Установка порта
PORTD матрицы резисторов сопротивлением 6,34 кОм.*/
#define ADCIHResistValDDR_5_84()
        MODIFY_REG(DDRD,((1<<PD6)),((1<<PD3)|(1<<PD4)|(1<<PD5)))    /*Установка порта
DDR матрицы резисторов сопротивлением 5,84 кОм.*/
```

```
#define ADCIHResistValPORTD_5_84()
	MODIFY_REG(PORTD,((1<<PD6)),((1<<PD3)|(1<<PD4)|(1<<PD5)))	/*Установка порта
PORTD матрицы резисторов сопротивлением 5,84 кОм.*/
#define ADCIHResistValDDR_5_46()
	SET_BITS(DDRD,((1<<PD3)|(1<<PD4)|(1<<PD5)|(1<<PD6)))		/*Установка порта
DDR матрицы резисторов сопротивлением 5,46 кОм.*/
#define ADCIHResistValPORTD_5_46()
	SET_BITS(PORTD,((1<<PD3)|(1<<PD4)|(1<<PD5)|(1<<PD6)))		/*Установка порта
PORTD матрицы резисторов сопротивлением 5,46 кОм.*/

#define ADCIHSampleStart()		SET_BIT(ADCSRA,ADSC) /*Функция запуска одной
выборки АЦП.*/

void ADCIHInit(void);				/*Функция инициализации АЦП.*/
uint16_t ADCIHHandler(void);		/*Функция обработки прерывания АЦП.*/

#endif /* ADCINTERFACEHALLIB_H_ */
```

Листинг файла ADCInterfaceLib.c

```
/*
 * Назначение: Файл модуля АЦП датчика.
 * Имя файла: ADCInterfaceLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 01.05.2015 г.
 * Список изменений:
 * 01.05.2015 г. − Исходная версия.
*/

#include "ADCInterfaceLib.h"

/*******************************************************************************
************
* Прототип: void ADCInitialization(void);
* Назначение: Функция инициализации АЦП.
* Аргументы: Нет.
* Возвращаемое значение: Нет.
* Описание: Функция инициализации АЦП.
*******************************************************************************
************/
void ADCInitialization(void)
{
	ADCSetResistorMatrixValue(ADC_RESISTOR_VALUE_0_00);
	ADCIHInit();
}

/*******************************************************************************
************
* Прототип: void ADCSetResistorMatrixValue(const enum ADCReistor_TypeDef);
* Назначение: Функция установки общего сопротивления матрицы резисторов.
* Аргументы: "ADCReistor" − общее сопротивление матрицы резисторов.
* Возвращаемое значение: Нет.
* Описание: Функция установки общего сопротивления матрицы резисторов.
*******************************************************************************
************/
void ADCSetResistorMatrixValue(const enum ADCReistor_TypeDef ADCReistor)
{
	static enum ADCReistor_TypeDef tmp=ADC_RESISTOR_VALUE_0_00;
	if (tmp!=ADCReistor)
	{
		tmp=ADCReistor;
		switch (tmp)
		{
			case ADC_RESISTOR_VALUE_82_00:
			{
				ADCIHResistValPORTD_82_00();
				ADCIHResistValDDR_82_00();
				break;
			}
			case ADC_RESISTOR_VALUE_39_00:
			{
```

```
        ADCIHResistValPORTD_39_00();
        ADCIHResistValDDR_39_00();
        break;
}
case ADC_RESISTOR_VALUE_26_43:
{
        ADCIHResistValPORTD_26_43();
        ADCIHResistValDDR_26_43();
        break;
}
case ADC_RESISTOR_VALUE_22_00:
{
        ADCIHResistValPORTD_22_00();
        ADCIHResistValDDR_22_00();
        break;
}
case ADC_RESISTOR_VALUE_17_35:
{
        ADCIHResistValPORTD_17_35();
        ADCIHResistValDDR_17_35();
        break;
}
case ADC_RESISTOR_VALUE_14_07:
{
        ADCIHResistValPORTD_14_07();
        ADCIHResistValDDR_14_07();
        break;
}
case ADC_RESISTOR_VALUE_12_01:
{
        ADCIHResistValPORTD_12_01();
        ADCIHResistValDDR_12_01();
        break;
}
case ADC_RESISTOR_VALUE_10_00:
{
        ADCIHResistValPORTD_10_00();
        ADCIHResistValDDR_10_00();
        break;
}
case ADC_RESISTOR_VALUE_8_91:
{
        ADCIHResistValPORTD_8_91();
        ADCIHResistValDDR_8_91();
        break;
}
case ADC_RESISTOR_VALUE_7_96:
{
        ADCIHResistValPORTD_7_96();
        ADCIHResistValDDR_7_96();
        break;
}
```

```
                        case ADC_RESISTOR_VALUE_7_25:
                        {
                                ADCIHResistValPORTD_7_25();
                                ADCIHResistValDDR_7_25();
                                break;
                        }
                        case ADC_RESISTOR_VALUE_6_88:
                        {
                                ADCIHResistValPORTD_6_88();
                                ADCIHResistValDDR_6_88();
                                break;
                        }
                        case ADC_RESISTOR_VALUE_6_34:
                        {
                                ADCIHResistValPORTD_6_34();
                                ADCIHResistValDDR_6_34();
                                break;
                        }
                        case ADC_RESISTOR_VALUE_5_84:
                        {
                                ADCIHResistValPORTD_5_84();
                                ADCIHResistValDDR_5_84();
                                break;
                        }
                        case ADC_RESISTOR_VALUE_5_46:
                        {
                                ADCIHResistValPORTD_5_46();
                                ADCIHResistValDDR_5_46();
                                break;
                        }
                        default:
                        {
                                ADCIHResistValDDR_0_00();
                                ADCIHResistValPORTD_0_00();
                                break;
                        }
                }
        }
}

/*****************************************************************************
************
 * Прототип функции: uint16_t ADCHandler(void);
 * Назначение: Функция обработки прерывания АЦП.
 * Аргументы: Нет.
 * Возвращаемые значения: Оцифрованное значение.
 * Описание: Функция обработки прерывания АЦП.

*****************************************************************************
***********/
uint16_t ADCHandler(void)
{
```

```
	register uint16_t tmp=ADCIHHandler();
	ADCSetResistorMatrixValue(ADC_RESISTOR_VALUE_0_00);
	return tmp;
}

/*****************************************************************************
************
 * Прототип функции: void ADCSampleStart(const enum ADCReistor_TypeDef);
 * Назначение: Функция запуска выборки АЦП.
 * Аргументы: "ADCReistor" − общее сопротивление матрицы резисторов.
 * Возвращаемые значения: Нет.
 * Описание: Функция запуска выборки АЦП.

*****************************************************************************
***********/
void ADCSampleStart(const enum ADCReistor_TypeDef ADCReistor)
{
	ADCSetResistorMatrixValue(ADCReistor);
	ADCIHSampleStart();
}
```

Листинг заголовочного файла ADCInterfaceLib.h

```
/*
 * Назначение: Заголовочный файл модуля АЦП датчика.
 * Имя файла: ADCInterfaceLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 01.05.2015 г.
 * Список изменений:
 * 01.05.2015 г. − Исходная версия.
 */

#ifndef ADCINTERFACELIB_H_
#define ADCINTERFACELIB_H_

#include "GenDefineLib.h"
#include "ADCInterfaceHALLib.h"

/*Возможные сопротивления матрицы резисторов.*/
enum ADCReistor_TypeDef
{
	ADC_RESISTOR_VALUE_0_00=0,		/*Общее сопротивление матрицы резисторов
0.00 кОм.*/
	ADC_RESISTOR_VALUE_82_00=1,		/*Общее сопротивление матрицы резисторов
82,00 кОм.*/
	ADC_RESISTOR_VALUE_39_00=2,		/*Общее сопротивление матрицы резисторов
39,00 кОм.*/
	ADC_RESISTOR_VALUE_26_43=3,		/*Общее сопротивление матрицы резисторов
26,43 кОм.*/
	ADC_RESISTOR_VALUE_22_00=4,		/*Общее сопротивление матрицы резисторов
22,00 кОм.*/
	ADC_RESISTOR_VALUE_17_35=5,		/*Общее сопротивление матрицы резисторов
17,35 кОм.*/
	ADC_RESISTOR_VALUE_14_07=6,		/*Общее сопротивление матрицы резисторов
14,07 кОм.*/
	ADC_RESISTOR_VALUE_12_01=7,		/*Общее сопротивление матрицы резисторов
12,01 кОм.*/
	ADC_RESISTOR_VALUE_10_00=8,		/*Общее сопротивление матрицы резисторов
10,00 кОм.*/
	ADC_RESISTOR_VALUE_8_91=9,		/*Общее сопротивление матрицы резисторов
8,91 кОм.*/
	ADC_RESISTOR_VALUE_7_96=10,		/*Общее сопротивление матрицы резисторов
7,96 кОм.*/
	ADC_RESISTOR_VALUE_7_25=11,		/*Общее сопротивление матрицы резисторов
7,25 кОм.*/
	ADC_RESISTOR_VALUE_6_88=12,		/*Общее сопротивление матрицы резисторов
6,88 кОм.*/
	ADC_RESISTOR_VALUE_6_34=13,		/*Общее сопротивление матрицы резисторов
6,34 кОм.*/
	ADC_RESISTOR_VALUE_5_84=14,		/*Общее сопротивление матрицы резисторов
5,84 кОм.*/
	ADC_RESISTOR_VALUE_5_46=15,		/*Общее сопротивление матрицы резисторов
5,46 кОм.*/
};
```

```
void ADCInitialization(void);
        /*Функция инициализации АЦП.*/
void ADCSetResistorMatrixValue(const enum ADCReistor_TypeDef);   /*Функция установки
общего сопротивления матрицы резисторов.*/
uint16_t ADCHandler(void);
        /*Функция обработки прерывания АЦП.*/
void ADCSampleStart(const enum ADCReistor_TypeDef);                    /*Функция
запуска выборки АЦП.*/

#endif /* ADCINTERFACELIB_H_ */
```

Листинг файла Config.h

```
/*
 * Назначение: Конфигурационный файл.
 * Имя файла: Config.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 18.04.2015 г.
 * Список изменений:
 * 18.04.2015 г. − Исходная версия.
 */

#ifndef CONFIG_H_
#define CONFIG_H_

#define F_CPU 1000000UL  /*Частота ЦПУ.*/

/*Настройка ОСРВ*/
#define CONFIG_MAX_TIME_COUNTER_VALUE 65535UL   /*Максимальное число
помещающееся в регистре таймера.*/
#define CONFIG_TIME_QUANT 0.002500                              /*Квант времени
планировщика в с.*/
#define CONFIG_MCU_FREQUENCY F_CPU                              /*Тактовая частота таймера
в Гц.*/

/*Настройка задач*/
#define CONFIG_LED_BLINK 1.0                  /*Отправка пакета данных (сек.).*/
#define CONFIG_START_ADC_SEMPLE 16.0    /*Запуск оцифровки датчика (Гц.).*/

#endif /* CONFIG_H_ */
```

Листинг файла GenDefineLib.h

```
/*
 * Назначение: Определение системных макросов, констант и определений.
 * Имя файла: GenDefineLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 16.10.2014 г.
 * Список изменений:
 * 16.10.2014 г. − Исходная версия.
 * 28.11.2014 г. − Добавлены определения байтов преамбулы и инструкция __nop().
 * 21.04.2015 г. − Добавлены макросы MODIFY_REG, SET_BITS, CLEAR_BITS.
 * 01.05.2015 г. − Добавлен макрос MASSIV_SIZE.
 */

#ifndef GENDEFINELIB_H_
#define GENDEFINELIB_H_

#include "Config.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define SET_BIT(REG,BIT)  (REG|=(1<<BIT))
#define CLEAR_BIT(REG,BIT)      (REG&=~(1<<BIT))
#define SET_BITS(REG,BIT)       (REG|=(BIT))
#define CLEAR_BITS(REG,BIT)     (REG&=~(BIT))
#define WRITE_REG(REG,BIT)      (REG=(BIT))
#define READ_REG(REG) (REG)
#define MODIFY_REG(REG,CLEARMASK,SETMASK)
        WRITE_REG((REG),(((READ_REG(REG))&(~(CLEARMASK)))|(SETMASK)))
#define TRUE (1==1)
#define FALSE (0==1)
#define SET TRUE
#define CLEAR FALSE
#define CHECKED_BIT(REG,BIT) (((REG)&(1<<BIT))?TRUE:FALSE)
#define HIGH(REG) ((REG)>>8)
#define LOW(REG) ((REG)&255)
#define BREAK while(1) /*Используется при отладке.*/
#define START_BYTE 0xAA /*Значение "Старт−байта в пакете.*/
#define STOP_BYTE 0x55 /*Значение "Стоп−байта в пакете.*/
#define PREAMBULE_BYTE_1 0xA5                   /*Байт преамбулы 1.*/
#define PREAMBULE_BYTE_2 0x5A                   /*Байт преамбулы 2.*/
#define MASSIV_SIZE(X)   (sizeof(X)/sizeof(X[0]))       /*Вычисление размера массива.*/

#define __nop() asm volatile ("NOP")        /*Вызов инструкции "NOP".*/

/*Общие состояния.*/
enum GeneralState_TypeDef
{
 GENERAL_STATE_TRUE, /*"Истина".*/
 GENERAL_STATE_FALSE, /*"Ложь".*/
 GENERAL_STATE_SET, /*"Установлен".*/
 GENERAL_STATE_CLEAR, /*"Снят".*/
```

```
 GENERAL_STATE_ENABLE, /*"Включено".*/
 GENERAL_STATE_DISABLE, /*"Выключено".*/
 GENERAL_STATE_NULL, /*Состояние не определено.*/
};

#endif /* GENDEFINELIB_H_ */
```

Листинг файла GeneralAlgorithmsLib.c

```
/*
 * Назначение: Файл модуля основных алгоритмов микропрограммы.
 * Имя файла: GeneralAlgorithmsLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 30.10.2014
 * Список изменений: 30.10.2014 − Исходная версия.
 * 01.05.2015 г. − Добавлена функция "GetPreambule".
 */

#include "GeneralAlgorithmsLib.h"

/*****************************************************************************
************
 * Прототип функции: float GAAverSumFunc(struct GAAverageBuffer_TypeDef*,const float);
 * Назначение: Функция вычисляет среднее значение суммы чисел поступающих на ее вход
методом \
 * скользящего окна.
 * Аргументы: "buffer" − буфер усредняемых значений. "number" − добавляемое к устреднению
число.
 * Возвращаемые значения: Вычисленная средняя величина.
 * Описание: Функция вычисляет среднее значение суммы чисел поступающих на ее вход
методом \
 * скользящего окна.

*****************************************************************************
***********/
float GAAverSumFunc(struct GAAverageBuffer_TypeDef* buffer,const float number)
{
	buffer−>Sum−=buffer−>buffer[buffer−>pBuffer];
	buffer−>buffer[buffer−>pBuffer++]=number;
	buffer−>Sum+=number;
	if(buffer−>pBuffer==buffer−>BufferLength)
	{
		buffer−>pBuffer=0;
	}
	return buffer−>Sum/buffer−>BufferLength;
}

/*****************************************************************************
************
 * Прототип функции: uint8_t GetXor(const uint8_t*,const uint8_t*);
 * Назначение: Функция получения байта контрольной суммы массива байт.
 * Аргументы: "firstElement" − первый элемент массива. "lastElement" − последний элемент \
 * массива.
 * Возвращаемые значения: Контрольная сумма массива байт типа "uint8_t".
 * Описание: Функция вычисляет контрольную сумму массива байт методом "исключающего
ИЛИ".

*****************************************************************************
***********/
uint8_t GetXor(const uint8_t* firstElement,const uint8_t* lastElement)
```

```
{
	register uint8_t tmp=0;
	if(firstElement!=lastElement)
	{
		tmp=*firstElement++;
		while(firstElement!=lastElement+1)
		{
			tmp^=*firstElement++;
		}
	}
 return tmp;
}

/*****************************************************************************
************
 * Прототип функции: uint8_t GetPreambule(void);
 * Назначение: Функция получения байта преамбулы.
 * Аргументы: Нет.
 * Возвращаемые значения: Байт преамбулы.
 * Описание: Функция получения байта преамбулы с чередующимся значением.

*****************************************************************************
***********/
uint8_t GetPreambule(void)
{
	static uint8_t tmp=PREAMBULE_BYTE_1;
	if(tmp==PREAMBULE_BYTE_1)
	{
		tmp=PREAMBULE_BYTE_2;
	}
	else
	{
		tmp=PREAMBULE_BYTE_1;
	}
	return tmp;
}

/*****************************************************************************
************
 * Прототип функции: uint16_t ByteToUint16_t(const uint8_t*);
 * Назначение: Функция получения числа типа "uint16_t" из массива байт.
 * Аргументы: "buffer" − указатель на первый элемент числа в буфере.
 * Возвращаемые значения: Число типа "uint16_t".
 * Описание: Функция получения числа типа "uint16_t" из массива байт.

*****************************************************************************
***********/
uint16_t ByteToUint16_t(const uint8_t* buffer)
{
	union Uint16_T_Convert_TypeDef tmp;
	tmp.byte[0]=buffer[0];
	tmp.byte[1]=buffer[1];
```

```
	return tmp.data;
}

/*****************************************************************************
************
 * Прототип функции: void Uint16_tToByte(uint8_t*,const uint16_t);
 * Назначение: Функция получения массива байт из числа типа "uint16_t".
 * Аргументы: "buffer" − указатель на первый элемент числа в буфере. "value" −
преобразовываемое число.
 * Возвращаемые значения: Нет.
 * Описание: Функция получения массива байт из числа типа "uint16_t".

*****************************************************************************
***********/
void Uint16_tToByte(uint8_t* buffer,const uint16_t value)
{
	union Uint16_T_Convert_TypeDef tmp;
	tmp.data=value;
	buffer[0]=tmp.byte[0];
	buffer[1]=tmp.byte[1];
}
```

Листинг заголовочного файла GeneralAlgorithmsLib.h

```
/*
 * Назначение: Заголовочный файл модуля основных алгоритмов микропрограммы.
 * Имя файла: GeneralAlgorithmsLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 30.10.2014
 * Список изменений: 30.10.2014 − Исходная версия.
 * 01.05.2015 г. − Добавлена функция "GetPreambule".
 */

#ifndef GENERALALGORITHMSLIB_H_
#define GENERALALGORITHMSLIB_H_

#include "GenDefineLib.h"

/*Буфер для усреднения значений методом скользящего окна.*/
struct GAAverageBuffer_TypeDef
{
 float* buffer; /*Указатель на массив буфера.*/
 const unsigned int BufferLength; /*Длина буфера.*/
 float Sum; /*Сумма элементов буфера.*/
 unsigned int pBuffer; /*Указатель на элемент буфера.*/
};

/*Объединение для преобразования полуслова.*/
union Uint16_T_Convert_TypeDef
{
	uint16_t data;				/*Слово.*/
	uint8_t byte[sizeof(uint16_t)];		/*Массив байт.*/
};

float GAAverSumFunc(struct GAAverageBuffer_TypeDef*,const float); /*Функция вычисляет
среднее значение суммы чисел поступающих на ее вход методом скользящего окна.*/
uint8_t GetXor(const uint8_t*,const uint8_t*); /*Функция получения байта контрольной суммы
массива байт.*/
uint8_t GetPreambule(void);
	/*Функция получения байта преамбулы.*/
uint16_t ByteToUint16_t(const uint8_t*);	/*Функция
получения числа типа "uint16_t" из массива байт.*/
void Uint16_tToByte(uint8_t*,const uint16_t);	/*Функция
получения массива байт из числа типа "uint16_t".*/

#endif /* GENERALALGORITHMSLIB_H_ */
```

Листинг заголовочного файла HIDInterfaceHALLib.h

```
/*
 * Назначение: Заголовочный файл модуля абстракции интерфейса взаимодействия с
пользователем.
 * Имя файла: HIDInterfaceHALLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 29.10.2014 г.
 * Список изменений:
 * 29.10.2014 г. − Исходная версия.
 */

#ifndef HIDINTERFACEHALLIB_H_
#define HIDINTERFACEHALLIB_H_

#include "GenDefineLib.h"

#define HIDIHLEDInit()      SET_BIT(DDRD,PD1)                /*Функция инициализации порта
светодиода.*/
#define HIDIHLEDSet() SET_BIT(PORTD,PD1)            /*Включить светодиод.*/
#define HIDIHLEDClear()    CLEAR_BIT(PORTD,PD1)  /*Погасить светодиод.*/

#endif /* HIDINTERFACEHALLIB_H_ */
```

Листинг файла HIDInterfaceLib.c

```
/*
 * Назначение: Файл модуля интерфейса взаимодействия с пользователем.
 * Имя файла: HIDInterfaceLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 29.10.2014 г.
 * Список изменений:
 * 29.10.2014 г. − Исходная версия.
 */

#include "HIDInterfaceLib.h"

enum GeneralState_TypeDef LEDstate=GENERAL_STATE_DISABLE; /*Начальное состояние
светодиода.*/

/*****************************************************************************
************
 * Прототип функции: void HIDIInitialization(void):
 * Назначение: Функция инициализации интерфейса с пользователем.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция инициализации интерфейса с пользователем.

*****************************************************************************
***********/
void HIDIInitialization(void)
{
 HIDIHLEDInit();
 HIDILEDSetState(LEDstate);
}

/*****************************************************************************
************
 * Прототип функции: void HIDILEDSetState(enum GeneralState_TypeDef);
 * Назначение: Функция задания режима работы светодиода "Включено"/"Выключено".
 * Аргументы: "state" − устанавливаемое состояние светодиодов.
 * Возвращаемые значения: Нет.
 * Описание: Функция задания режима работы светодиода "Включено"/"Выключено".

*****************************************************************************
***********/
void HIDILEDSetState(enum GeneralState_TypeDef state)
{
 if(state==GENERAL_STATE_ENABLE)
 {
 LEDstate=GENERAL_STATE_ENABLE;
 HIDIHLEDSet();
 }
 else
 {
 LEDstate=GENERAL_STATE_DISABLE;
 HIDIHLEDClear();
```

```
 }
}

/*****************************************************************************
************
 * Прототип функции: void HIDILEDChangeState(void);
 * Назначение: Функция изменения состояния светодиода на противоположенное.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция изменения состояния светодиода на противоположенное.

*****************************************************************************
***********/
void HIDILEDChangeState(void)
{
 if(LEDstate!=GENERAL_STATE_ENABLE)
 {
 HIDILEDSetState(GENERAL_STATE_ENABLE);
 }
 else
 {
 HIDILEDSetState(GENERAL_STATE_DISABLE);
 }
}

/*****************************************************************************
************
 * Прототип функции: enum GeneralState_TypeDef HIDILEDGetState(void);
 * Назначение: Функция получения состояния светодиода.
 * Аргументы: Нет.
 * Возвращаемые значения: Состояние светодиода.
 * Описание: Функция получения состояния светодиода.

*****************************************************************************
***********/
enum GeneralState_TypeDef HIDILEDGetState(void)
{
 return LEDstate;
}
```

Листинг заголовочного файла HIDInterfaceLib.h

```
/*
 * Назначение: Заголовочный файл модуля интерфейса взаимодействия с пользователем.
 * Имя файла: HIDInterfaceLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 29.10.2014 г.
 * Список изменений:
 * 29.10.2014 г. − Исходная версия.
 */

#ifndef HIDINTERFACELIB_H_
#define HIDINTERFACELIB_H_

#include "HIDInterfaceHALLib.h"

void HIDIInitialization(void); /*Функция инициализации интерфейса с пользователем.*/
void HIDILEDSetState(enum GeneralState_TypeDef); /*Функция задания режима работы
светодиода "Включено"/"Выключено".*/
void HIDILEDChangeState(void); /*Функция изменения состояния светодиода на
противоположенное.*/
enum GeneralState_TypeDef HIDILEDGetState(void); /*Функция получения состояния
светодиода.*/

#endif /* HIDINTERFACELIB_H_ */
```

Листинг файла CaudaScopeSensor.c

```
/*
 * Назначение: Основной файл микропрограммы.
 * Имя файла: CaudaScopeSensor.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 30.04.2015 г.
 * Список изменений: 30.04.2015 г. − Исходная версия.
 */

#include "GenDefineLib.h"
#include "RTOSShedulerLib.h"

ISR(TIMER1_COMPA_vect)
{
	RTOSTaskCounter();
}

ISR(INT0_vect)
{
	RTOSIRQHandler(RTOS_IRQ_HW_RF_MODULE);
}

ISR(ADC_vect)
{
	RTOSIRQHandler(RTOS_IRQ_HW_ADC);
}

/*****************************************************************************
************
 * Прототип функции: int main(void)
 * Назначение: Главная функция микропрограммы.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Главная функция микропрограммы.

*****************************************************************************
***********/
int main(void)
{
	RTOSHardwareInitialization();
	while(1)
	{
		RTOSTaskHandler();
	}
}
```

Листинг файла RFInterfaceHALLib.c

```
/*
 * Назначение: Файл модуля абстракции радиоинтерфейса от аппаратуры.
 * Имя файла: RFInterfaceHALLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 23.10.2014 г.
 * Список изменений:
 * 23.10.2014 г. − Исходная версия.
 */

#include "RFInterfaceHALLib.h"

/*Блок объявления внутренних функций.*/
void RFIH_Port_Init(void); /*Функция настройки портов для взаимодействия с радиомодулем.*/
void RFIH_IRQ_Init(void); /*Функция настройки запроса прерывания от радиомодуля.*/
void RFIH_SPI_Init(void); /*Функция настройки интерфейса SPI.*/

/*****************************************************************************
************
 * Прототип функции: void RFIH_Hardware_Init(void);
 * Назначение: Функция настройки аппаратного обеспечения для взаимодействия с
радиомодулем.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки аппаратного обеспечения для взаимодействия с
радиомодулем.

*****************************************************************************
***********/
void RFIH_Hardware_Init(void)
{
 RFIH_Port_Init();
 RFIH_SPI_Init();
 RFIH_IRQ_Init();
}

/*****************************************************************************
************
 * Прототип функции: void RFIH_Port_Init(void);
 * Назначение: Функция настройки портов для взаимодействия с радиомодулем.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки портов для взаимодействия с радиомодулем.

*****************************************************************************
***********/
void RFIH_Port_Init(void)
{
	MODIFY_REG(RF_PORT,(1<<VCC),(1<<CE));
	SET_BITS(RF_DDR,((1<<VCC)|(1<<CE)));
	SET_BITS(SPI_PORT,((1<<SS)|(1<<MISO)|(1<<MOSI)|(1<<SCK)));
	SET_BITS(SPI_DDR,((1<<SS)|(1<<MOSI)|(1<<SCK)|(1<<MISO)));
```

```
}

/*****************************************************************************
************
 * Прототип функции: void RFIH_IRQ_Init(void);
 * Назначение: Функция настройки запроса прерывания от радиомодуля.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки запроса прерывания от радиомодуля.

*****************************************************************************
***********/
void RFIH_IRQ_Init(void)
{
	CLEAR_BITS(MCUCR,((1<<ISC01)|(1<<ISC00)));
	SET_BIT(GICR,INT0);
}

/*****************************************************************************
************
 * Прототип функции: void RFIH_SPI_Init(void);
 * Назначение: Функция настройки интерфейса SPI.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настраивает интерфейс SPI в режиме "Ведущего".

*****************************************************************************
***********/
void RFIH_SPI_Init(void)
{
	MODIFY_REG(SPCR,((1<<SPIE)|(1<<DORD)|(1<<CPOL)|(1<<CPHA)|(1<<SPR1)|(1<<SP
R0)),((1<<SPE)|(1<<MSTR)));
	SET_BIT(SPSR,SPI2X);
}

/*****************************************************************************
************
 * Прототип функции: void RFIH_Powered(void);
 * Назначение: Функция включения подачи питания на радиомодуль.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция включения подачи питания на радиомодуль.

*****************************************************************************
***********/
void RFIH_Powered(void)
{
	SET_BIT(RF_PORT,VCC);
	_delay_ms(12);
}
```

```
/*****************************************************************************
************
 * Прототип функции: void RFIH_Unpowered(void);
 * Назначение: Функция выключения подачи питания на радиомодуль.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция выключения подачи питания на радиомодуль.

*****************************************************************************
***********/
void RFIH_Unpowered(void)
{
	CLEAR_BIT(RF_PORT,VCC);
	_delay_ms(12);
}
```

Листинг заголовочного файла RFInterfaceHALLib.h

```
/*
 * Назначение: Заголовочный файл модуля абстракции радиоинтерфейса от аппаратуры.
 * Имя файла: RFInterfaceHALLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 23.10.2014 г.
 * Список изменений:
 * 23.10.2014 г. − Исходная версия.
 */

#ifndef RFINTERFACEHALLIB_H_
#define RFINTERFACEHALLIB_H_

#include "GenDefineLib.h"

/*номера потров радиоинтерфейса.*/
#define VCC   0
#define CE     1

/*Номера портов интерфейса SPI.*/
#define SS              4
#define MOSI 5
#define MISO 6
#define SCK             7

#define SPI_DDR DDRB      /*Порт интерфейса SPI.*/
#define SPI_PORT PORTB   /*Порт интерфейса SPI.*/

#define RF_PORT PORTB   /*Выводы радиоинтерфейса.*/
#define RF_DDR DDRB                /*Порт радиоинтерфейса.*/

#define RFIH_CSN_Pin_Clear()                        CLEAR_BIT(RF_PORT,SS)
        /*Установить низкий логический уровень на выводе CSN.*/
#define RFIH_CSN_Pin_Set()                                  SET_BIT(RF_PORT,SS)
        /*Установить высокий логический уровень на выводе CSN.*/
#define RFIH_CE_Pin_Set()                           SET_BIT(RF_PORT,CE)
        /*Установить высокий логический уровень на выводе CE.*/
#define RFIH_CE_Pin_Clear()                                 CLEAR_BIT(RF_PORT,CE)
        /*Установить низкий логический уровень на выводе CE.*/
#define RFIH_Write_Byte(byte)                       WRITE_REG(SPDR,(uint8_t)byte)  /*Записать
байт.*/
#define RFIH_Read_Byte()                            (uint8_t)SPDR
        /*Считать байт.*/
#define RFIH_Check_End_Transaction()    CHECKED_BIT(SPSR,SPIF)
        /*Проверить завершение передачи.*/
#define RFIH_IRQ_Acknowledgement()              SET_BIT(GIFR,INTF0)
        /*Подтверждение обработки запроса прерывания.*/

void RFIH_Hardware_Init(void); /*Функция настройки аппаратного обеспечения для
взаимодействия с радиомодулем.*/
void RFIH_Powered(void);            /*Функция включения подачи питания на радиомодуль.*/
```

```
void RFIH_Unpowered(void);                    /*Функция выключения подачи питания на
радиомодуль.*/

#endif /* RFINTERFACEHALLIB_H_ */
```

Листинг файла RFInterfaceLib.c

```
/*
 * Назначение: Файл модуля радиоинтерфейса.
 * Имя файла: RFInterfaceLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 23.10.2014 г.
 * Список изменений:
 * 23.10.2014 г. − Исходная версия.
 */

#include "RFInterfaceLib.h"

uint8_t RFI_Device_Address[]={0x40,0x50,0x60,0x70,0x80};      /*Адрес устройства.*/

struct RFIR_Registers_TypeDef RF_Device_Register;                        /*Регистры устройства.*/

/*****************************************************************************
************
 * Прототип функции: void RFI_Initialization(void);
 * Назначение: Функция настройки радиомодуля.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки радиомодуля.

*****************************************************************************
***********/
void RFI_Initialization(void)
{
	RFIH_Hardware_Init();
	RFIH_Unpowered();
	RFIH_Powered();
	RFI_Read_Device_Registers();
	/*Общие настройки.*/

	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P0=CLEAR;
	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P1=CLEAR;
	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P2=CLEAR;
	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P3=CLEAR;
	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P4=CLEAR;
	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P5=CLEAR;
	RFIR_Write_Register(RFIR_REG_TYPE_EN_AA,RF_Device_Register.EN_AA_Register.byt
e);

	RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P0=SET
;
	RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P1=CLE
AR;
	RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P2=CLE
AR;
	RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P3=CLE
AR;
```

```
RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P4=CLE
AR;
RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P5=CLE
AR;
RFIR_Write_Register(RFIR_REG_TYPE_EN_RXADDR,RF_Device_Register.EN_RXADDR
_Register.byte);

RF_Device_Register.SETUP_AW_Register.RFIR_SETUP_AW_ByteField.AW=(5−2);
RFIR_Write_Register(RFIR_REG_TYPE_SETUP_AW,RF_Device_Register.SETUP_AW_Re
gister.byte);

RF_Device_Register.SETUP_RETR_Register.RFIR_SETUP_RETR_ByteField.ARD=(750/25
0−1);
RF_Device_Register.SETUP_RETR_Register.RFIR_SETUP_RETR_ByteField.ARC=10;
RFIR_Write_Register(RFIR_REG_TYPE_SETUP_RETR,RF_Device_Register.SETUP_RET
R_Register.byte);

RF_Device_Register.RF_CH_Register.RFIR_RF_CH_ByteField.RF_CH=40;
RFIR_Write_Register(RFIR_REG_TYPE_RF_CH,RF_Device_Register.RF_CH_Register.byte
);

RF_Device_Register.RF_SETUP_Register.RFIR_RF_SETUP_ByteField.RF_DR_HIGH=CLE
AR;
RF_Device_Register.RF_SETUP_Register.RFIR_RF_SETUP_ByteField.RF_DR_LOW=SET;
RF_Device_Register.RF_SETUP_Register.RFIR_RF_SETUP_ByteField.RF_PWR=3;
RFIR_Write_Register(RFIR_REG_TYPE_RF_SETUP,RF_Device_Register.RF_SETUP_Regi
ster.byte);

RF_Device_Register.DYNPD_Register.byte=0;
RFIR_Write_Register(RFIR_REG_TYPE_DYNPD,RF_Device_Register.DYNPD_Register.by
te);

RF_Device_Register.FEATURE_Register.byte=0;
RFIR_Write_Register(RFIR_REG_TYPE_FEATURE,RF_Device_Register.FEATURE_Regist
er.byte);

RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.EN_CRC=SET;
RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.CRCO=SET;

RFIR_Flush_Buffer(RFIR_COM_TYPE_FLUSH_TX);
RFIR_Flush_Buffer(RFIR_COM_TYPE_FLUSH_RX);

/*Настройка передатчика.*/
RF_Device_Register.RFIR_RX_PW_P0_Register.RFIR_RX_PW_PX_ByteField.RX_PW_PX
=8;
RFIR_Write_Register(RFIR_REG_TYPE_RX_PW_P0,RF_Device_Register.RFIR_RX_PW_
P0_Register.byte);
RFIR_Write_Buffer(RFIR_REG_TYPE_TX_ADDR,RFI_Device_Address,sizeof(RFI_Device
_Address)/sizeof(RFI_Device_Address[0]));
RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.MASK_RX_DR=SET;
RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.MASK_TX_DS=CLEAR;
RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.MASK_MAX_RT=SET;
```

```
	RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.PWR_UP=SET;
	RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.PRIM_RX=CLEAR;
	RFIR_Write_Register(RFIR_REG_TYPE_CONFIG,RF_Device_Register.CONFIG_Register.b
yte);

	_delay_ms(3);
}

/*****************************************************************************
************
 * Прототип функции: int RFI_IRQ_Handler(void);
 * Назначение: Функция обработки прерывания от радиомодуля.
 * Аргументы: Нет.
 * Возвращаемые значения: Количество принятых байт.
 * Описание: Функция обработки прерывания от радиомодуля.

*****************************************************************************
***********/
int RFI_IRQ_Handler(void)
{
	register int tmp=0;
	RF_Device_Register.STATUS_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_STA
TUS);
	if(RF_Device_Register.STATUS_Register.RFIR_STATUS_ByteField.TX_DS==SET)
	{
		RFIR_Flush_Buffer(RFIR_COM_TYPE_FLUSH_TX);
	}
	RFIR_Write_Register(RFIR_REG_TYPE_STATUS,RF_Device_Register.STATUS_Register.
byte);
	RFIH_IRQ_Acknowledgement();
	_delay_us(150);
	return tmp;
}

/*****************************************************************************
************
 * Прототип функции: void RFI_Read_Device_Registers(void);
 * Назначение: Функция опроса регистров радиомодуля.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция опроса регистров радиомодуля.

*****************************************************************************
***********/
void RFI_Read_Device_Registers(void)
{
	RF_Device_Register.CONFIG_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_CON
FIG);
	RF_Device_Register.EN_AA_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_EN_A
A);
	RF_Device_Register.EN_RXADDR_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_
EN_RXADDR);
```

```
	RF_Device_Register.SETUP_AW_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_S
ETUP_AW);
	RF_Device_Register.SETUP_RETR_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_
SETUP_RETR);
	RF_Device_Register.RF_CH_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RF_C
H);
	RF_Device_Register.RF_SETUP_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RF
_SETUP);
	RF_Device_Register.STATUS_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_STA
TUS);
	RF_Device_Register.OBSERVE_TX_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE
_OBSERVE_TX);
	RF_Device_Register.RPD_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RPD);
	RF_Device_Register.RFIR_RX_PW_P0_Register.byte=RFIR_Read_Register(RFIR_REG_TY
PE_RX_PW_P0);
	RF_Device_Register.RFIR_RX_PW_P1_Register.byte=RFIR_Read_Register(RFIR_REG_TY
PE_RX_PW_P1);
	RF_Device_Register.RFIR_RX_PW_P2_Register.byte=RFIR_Read_Register(RFIR_REG_TY
PE_RX_PW_P2);
	RF_Device_Register.RFIR_RX_PW_P3_Register.byte=RFIR_Read_Register(RFIR_REG_TY
PE_RX_PW_P3);
	RF_Device_Register.RFIR_RX_PW_P4_Register.byte=RFIR_Read_Register(RFIR_REG_TY
PE_RX_PW_P4);
	RF_Device_Register.RFIR_RX_PW_P5_Register.byte=RFIR_Read_Register(RFIR_REG_TY
PE_RX_PW_P5);
	RF_Device_Register.FIFO_STATUS_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE
_FIFO_STATUS);
	RF_Device_Register.DYNPD_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_DYNP
D);
	RF_Device_Register.FEATURE_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_FE
ATURE);
}

/*****************************************************************************
************
 * Прототип функции: void RFI_Send_Buffer(const uint8_t*,const uint8_t);
 * Назначение: Генерация передачи данных по радиоканалу.
 * Аргументы: "buffer" − указатель на буфер с данными. "buffer_length" − размер буфера
данных.
 * Возвращаемые значения:
 * Описание: Генерирует передачу данных по радиоканалу.

*****************************************************************************
***********/
void RFI_Send_Buffer(const uint8_t* buffer,const uint8_t buffer_length)
{
 RFIR_Write_Buffer(RFIR_COM_TYPE_W_TX_PAYLOAD,buffer,buffer_length);
 RFIR_Run_Write(buffer_length);
}
```

Листинг заголовочного файла RFInterfaceLib.h

```
/*
 * Назначение: Заголовочный файл модуля радиоинтерфейса.
 * Имя файла: RFInterfaceLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 23.10.2014 г.
 * Список изменений:
 * 23.10.2014 г. − Исходная версия.
 */

#ifndef RFINTERFACELIB_H_
#define RFINTERFACELIB_H_

#include "GenDefineLib.h"
#include "GeneralAlgorithmsLib.h"
#include "RFInterfaceHALLib.h"
#include "RFInterfaceRELLib.h"

void RFI_Initialization(void);                                  /*Функция настройки
радиомодуля. */
int RFI_IRQ_Handler(void);                                      /*Функция обработки
прерывания от радиомодуля.*/
void RFI_Read_Device_Registers(void);                           /*Функция опроса регистров
радиомодуля.*/
void RFI_Send_Buffer(const uint8_t*,const uint8_t);         /*Генерация передачи данных по
радиоканалу.*/

#endif /* RFINTERFACELIB_H_ */
```

Листинг файла RFInterfaceRELLib.c

```
/*
 * Назначение: Файл модуля доступа к регистрам радиоинтерфейса.
 * Имя файла: RFInterfaceRELLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 05.11.2014 г.
 * Список изменений:
 * 05.11.2014 г. − Исходная версия.
 */

#include "RFInterfaceRELLib.h"

/*Блок объявления внутренних функций.*/
uint8_t RFIR_WR_Byte(const uint8_t); /*Функция передачи/получения байта данных в/от
радиомодуля.*/

/*****************************************************************************
************
 * Прототип функции: uint8_t RFIR_WR_Byte(const uint8_t);
 * Назначение: Функция передачи/получения байта данных в/от радиомодуля.
 * Аргументы: "byte" − байт данных для передачи радиомодулю.
 * Возвращаемые значения: Принятый от радиомодуля байт данных.
 * Описание: Функция передачи/получения байта данных в/от радиомодуля.

*****************************************************************************
***********/
uint8_t RFIR_WR_Byte(const uint8_t byte)
{
	RFIH_Write_Byte(byte);
	while(!RFIH_Check_End_Transaction());
	return RFIH_Read_Byte();
}

/*Основные функции для работы с модулем.*/

/*****************************************************************************
************
 * Прототип функции: uint8_t RFIR_Read_Register(const uint8_t);
 * Назначение: Функция чтения из регистра.
 * Аргументы: "reg" − регистр.
 * Возвращаемые значения: Состояние регистра радомодуля.
 * Описание: Функция чтения данных из регистра радиомодуля.

*****************************************************************************
***********/
uint8_t RFIR_Read_Register(const uint8_t reg)
{
	register uint8_t tmp;
	RFIH_CSN_Pin_Clear();
	RFIR_WR_Byte(RFIR_COM_TYPE_R_REGISTER|reg);
	tmp=RFIR_WR_Byte(0);
	RFIH_CSN_Pin_Set();
```

```
	return tmp;
}

/*****************************************************************************
************
 * Прототип функции: void RFIR_Write_Register(const uint8_t,const uint8_t);
 * Назначение: Функция записи в регистр.
 * Аргументы: "reg" − регистр. "value" − записываемое значение.
 * Возвращаемые значения: Нет.
 * Описание: Функция записи данных в регистр радиомодуля.

*****************************************************************************
***********/
void RFIR_Write_Register(const uint8_t reg,const uint8_t value)
{
	RFIH_CSN_Pin_Clear();
	RFIR_WR_Byte(RFIR_COM_TYPE_W_REGISTER|reg);
	RFIR_WR_Byte(value);
	RFIH_CSN_Pin_Set();
}

/*****************************************************************************
************
 * Прототип функции: void RFIR_Read_Buffer(const uint8_t,uint8_t*,const uint8_t);
 * Назначение: Функция чтения массива данных из радиомодуля.
 * Аргументы: "reg" − регистр. "buffer" − буффер для прочитанной информации. \
 * "byte_num" − количество байт для чтения.
 * Возвращаемые значения: Нет.
 * Описание: Функция чтения массива данных из радиомодуля.

*****************************************************************************
***********/
void RFIR_Read_Buffer(const uint8_t reg,uint8_t* buffer,const uint8_t byte_num)
{
	register uint8_t i;
	RFIH_CSN_Pin_Clear();
	RFIR_WR_Byte(RFIR_COM_TYPE_R_REGISTER|reg);
	for(i=0;i<byte_num;i++)
	{
		buffer[i]=RFIR_WR_Byte(0);
	}
	RFIH_CSN_Pin_Set();
}

/*****************************************************************************
************
 * Прототип функции: void RFIR_Write_Buffer(const uint8_t,const uint8_t*,const uint8_t);
 * Назначение: Функция записи массива данных в радиомодуль.
 * Аргументы: "reg" − регистр. "buffer" − буффер для прочитанной информации. \
 * "byte_num" − количество байт для чтения.
 * Возвращаемые значения: Нет.
 * Описание: Функция записи массива данных в радиомодуль.
```

```
*****************************************************************************
***********/
void RFIR_Write_Buffer(const uint8_t reg,const uint8_t* buffer,const uint8_t byte_num)
{
	register uint8_t i;
	RFIH_CSN_Pin_Clear();
	RFIR_WR_Byte(RFIR_COM_TYPE_W_REGISTER|reg);
	for(i=0;i<byte_num;i++)
	{
		RFIR_WR_Byte(buffer[i]);
	}
	RFIH_CSN_Pin_Set();
}

/*Основные комманды для работы с модулем.*/

/*****************************************************************************
************
 * Прототип функции: uint8_t RFIR_Get_Status(void);
 * Назначение: Функция считывания регистра состояния радиомодуля.
 * Аргументы: Нет.
 * Возвращаемые значения: Значение регистра состояния радиомодуля.
 * Описание: Функция считывания регистра состояния радиомодуля.

*****************************************************************************
***********/
uint8_t RFIR_Get_Status(void)
{
	register uint8_t tmp;
	RFIH_CSN_Pin_Clear();
	tmp=RFIR_WR_Byte(RFIR_COM_TYPE_NOP);
	RFIH_CSN_Pin_Set();
	return tmp;
}

/*****************************************************************************
************
 * Прототип функции: void RFIR_Flush_Buffer(const enum RFIR_Command_TypeDef);
 * Назначение: Функция очистки буферов приема и передачи радиомодуля.
 * Аргументы: "connand" − комманда.
 * Возвращаемые значения: Нет.
 * Описание: Функция очистки буферов приема и передачи радиомодуля.

*****************************************************************************
***********/
void RFIR_Flush_Buffer(const enum RFIR_Command_TypeDef connand)
{
	RFIH_CSN_Pin_Clear();
	if(connand==RFIR_COM_TYPE_FLUSH_TX)
	{
		RFIR_WR_Byte(RFIR_COM_TYPE_FLUSH_TX);
```

```
	}
	else
	{
		RFIR_WR_Byte(RFIR_COM_TYPE_FLUSH_RX);
	}
	RFIH_CSN_Pin_Set();
}

/*****************************************************************************
************
 * Прототип функции: void RFIR_Run_Write(const uint8_t);
 * Назначение: Функция запуска передачи данных.
 * Аргументы: "num" − количество передач.
 * Возвращаемые значения: Нет.
 * Описание: Функция запуска передачи данных.

*****************************************************************************
***********/
void RFIR_Run_Write(const uint8_t num)
{
	register uint8_t i;
	for(i=0;i<num;i++)
	{
		RFIH_CE_Pin_Clear();
		_delay_us(20);
		RFIH_CE_Pin_Set();
		_delay_us(20);
	}
}
```

Листинг заголовочного файла RFInterfaceRELLib.h

```
/*
 * Назначение: Заголовочный файл модуля доступа к регистрам радиоинтерфейса.
 * Имя файла: RFInterfaceRELLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 05.11.2014 г.
 * Список изменений:
 * 05.11.2014 г. − Исходная версия.
 */

#ifndef RFINTERFACERELLIB_H_
#define RFINTERFACERELLIB_H_

#include "RFInterfaceHALLib.h"

/*Команды для взаимодействия с радомодулем.*/
enum RFIR_Command_TypeDef
{
 RFIR_COM_TYPE_R_REGISTER=0x00, /*Прочитать из регистра.*/
 RFIR_COM_TYPE_W_REGISTER=0x20, /*Записать в регистр.*/
 RFIR_COM_TYPE_R_RX_PAYLOAD=0x61, /*Прочитать полученную полезную нагрузку.*/
 RFIR_COM_TYPE_W_TX_PAYLOAD=0xA0, /*Записать передаваемую полезную нагрузку.*/
 RFIR_COM_TYPE_FLUSH_TX=0xE1, /*Очистить буфер передачи.*/
 RFIR_COM_TYPE_FLUSH_RX=0xE2, /*Очистить буфер приема.*/
 RFIR_COM_TYPE_REUSE_TX_PL=0xE3, /*Повторить передачу полезной нагрузки из буфера
передачи.*/
 RFIR_COM_TYPE_R_RX_PL_WID=0x60, /*Считать принятую полезную нагрузку с вершины
стека.*/
 RFIR_COM_TYPE_W_ACK_PAYLOAD=0xA8, /*Записать полезную нагрузку передаваемую в
фазе подтверждения полученного пакета.*/
 RFIR_COM_TYPE_W_TX_PAYLOAD_NOACK=0xB0, /*Записать полезную нагузку с отменой
передачи полезной нагрузки в фазе подтверждения полученного пакета.*/
 RFIR_COM_TYPE_NOP=0xFF, /*Прочитать сегистр состояния радиомодуля.*/
};

/**/
enum RFIR_Register_TypeDef
{
 RFIR_REG_TYPE_CONFIG=0x00, /*Адрес регистра "CONFIG".*/
 RFIR_REG_TYPE_EN_AA=0x01, /*Адрес регистра "EN_AA".*/
 RFIR_REG_TYPE_EN_RXADDR=0x02, /*Адрес регистра "RXADDR".*/
 RFIR_REG_TYPE_SETUP_AW=0x03, /*Адрес регистра "SETUP_AW".*/
 RFIR_REG_TYPE_SETUP_RETR=0x04, /*Адрес регистра "SETUP_RETR".*/
 RFIR_REG_TYPE_RF_CH=0x05, /*Адрес регистра "RF_CH".*/
 RFIR_REG_TYPE_RF_SETUP=0x06, /*Адрес регистра "RF_SETUP".*/
 RFIR_REG_TYPE_STATUS=0x07, /*Адрес регистра "STATUS".*/
 RFIR_REG_TYPE_OBSERVE_TX=0x08, /*Адрес регистра "OBSERVE_TX".*/
 RFIR_REG_TYPE_RPD=0x09, /*Адрес регистра "RPD".*/
 RFIR_REG_TYPE_RX_ADDR_P0=0x0A, /*Адрес регистра "RX_ADDR_P0".*/
 RFIR_REG_TYPE_RX_ADDR_P1=0x0B, /*Адрес регистра "RX_ADDR_P1".*/
 RFIR_REG_TYPE_RX_ADDR_P2=0x0C, /*Адрес регистра "RX_ADDR_P2".*/
 RFIR_REG_TYPE_RX_ADDR_P3=0x0D, /*Адрес регистра "RX_ADDR_P3".*/
```

```
 RFIR_REG_TYPE_RX_ADDR_P4=0x0E, /*Адрес регистра "RX_ADDR_P4".*/
 RFIR_REG_TYPE_RX_ADDR_P5=0x0F, /*Адрес регистра "RX_ADDR_P5".*/
 RFIR_REG_TYPE_TX_ADDR=0x10, /*Адрес регистра "TX_ADDR".*/
 RFIR_REG_TYPE_RX_PW_P0=0x11, /*Адрес регистра "RX_PW_P0".*/
 RFIR_REG_TYPE_RX_PW_P1=0x12, /*Адрес регистра "RX_PW_P1".*/
 RFIR_REG_TYPE_RX_PW_P2=0x13, /*Адрес регистра "RX_PW_P2".*/
 RFIR_REG_TYPE_RX_PW_P3=0x14, /*Адрес регистра "RX_PW_P3".*/
 RFIR_REG_TYPE_RX_PW_P4=0x15, /*Адрес регистра "RX_PW_P4".*/
 RFIR_REG_TYPE_RX_PW_P5=0x16, /*Адрес регистра "RX_PW_P5".*/
 RFIR_REG_TYPE_FIFO_STATUS=0x17, /*Адрес регистра "FIFO_STATUS".*/
 RFIR_REG_TYPE_DYNPD=0x1C, /*Адрес регистра "DYNPD".*/
 RFIR_REG_TYPE_FEATURE=0x1D, /*Адрес регистра "FEATURE".*/
};

/*Структура регистра "CONFIG".*/
struct RFIR_CONFIG_ByteField_TypeDef
{
 uint8_t PRIM_RX:1; /*Бит режима работы радиомодуля приемник/передатчик.*/
 uint8_t PWR_UP:1; /*Включение питания радиомодуля.*/
 uint8_t CRCO:1; /*Контрольная сумма длиной 1 или 2 байта.*/
 uint8_t EN_CRC:1; /*Включение контрольной суммы.*/
 uint8_t MASK_MAX_RT:1; /*Включение/выключение прерывания MAX_RT.*/
 uint8_t MASK_TX_DS:1; /*Включение/выключение прерывания TX_DS.*/
 uint8_t MASK_RX_DR:1; /*Включение/выключение прерывания RX_DR.*/
 uint8_t RESERVED:1; /*Зарезервировано.*/
};

/*Структура регистра "EN_AA".*/
struct RFIR_EN_AA_ByteField_TypeDef
{
 uint8_t ENAA_P0:1; /*Включение/выключение автоматического подтверждения принятого
пакета логического канала 0.*/
 uint8_t ENAA_P1:1; /*Включение/выключение автоматического подтверждения принятого
пакета логического канала 1.*/
 uint8_t ENAA_P2:1; /*Включение/выключение автоматического подтверждения принятого
пакета логического канала 2.*/
 uint8_t ENAA_P3:1; /*Включение/выключение автоматического подтверждения принятого
пакета логического канала 3.*/
 uint8_t ENAA_P4:1; /*Включение/выключение автоматического подтверждения принятого
пакета логического канала 4.*/
 uint8_t ENAA_P5:1; /*Включение/выключение автоматического подтверждения принятого
пакета логического канала 5.*/
 uint8_t RESERVED:2; /*Зарезервировано.*/
};

/*Структура регистра "EN_RXADDR".*/
struct RFIR_EN_RXADDR_ByteField_TypeDef
{
 uint8_t ERX_P0:1; /*Включение/выключение логического канала 0.*/
 uint8_t ERX_P1:1; /*Включение/выключение логического канала 1.*/
 uint8_t ERX_P2:1; /*Включение/выключение логического канала 2.*/
 uint8_t ERX_P3:1; /*Включение/выключение логического канала 3.*/
```

```
 uint8_t ERX_P4:1; /*Включение/выключение логического канала 4.*/
 uint8_t ERX_P5:1; /*Включение/выключение логического канала 5.*/
 uint8_t RESERVED:2; /*Зарезервировано.*/
};

/*Структура регистра "SETUP_AW".*/
struct RFIR_SETUP_AW_ByteField_TypeDef
{
 uint8_t AW:2; /*Ширина поля адреса.*/
 uint8_t RESERVED:6; /*Зарезервировано.*/
};

/*Структура регистра "SETUP_RETR".*/
struct RFIR_SETUP_RETR_ByteField_TypeDef
{
 uint8_t ARC:4; /*Счетчик автоматических повторных передачь.*/
 uint8_t ARD:4; /*Время автоматической задержки передачи.*/
};

/*Структура регистра "RF_CH".*/
struct RFIR_RF_CH_ByteField_TypeDef
{
 uint8_t RF_CH:7; /*Радиоканал.*/
 uint8_t RESERVED:1; /*Зарезервировано.*/
};

/*Структура регистра "RF_SETUP".*/
struct RFIR_RF_SETUP_ByteField_TypeDef
{
 uint8_t Obsolete:1; /*Не используется.*/
 uint8_t RF_PWR:2; /*Мощность выходного сигнала.*/
 uint8_t RF_DR_HIGH:1; /*Скорость 1 или 2 МБит/с.*/
 uint8_t PLL_LOCK:1; /*Включение блокировки ФАПЧ. Только для тестового режима.*/
 uint8_t RF_DR_LOW:1; /*Скорость 250 кБит/с.*/
 uint8_t RESERVED:1; /*Зарезервировано.*/
 uint8_t CONT_WAVE:1; /*Включение несущей. Только для тестового режима.*/
};

/*Структура регистра "STATUS".*/
struct RFIR_STATUS_ByteField_TypeDef
{
 uint8_t TX_FULL:1; /*Флаг заполнения буфера передатчика.*/
 uint8_t RX_P_NO:3; /*Номер логического канала в котором приняты данные.*/
 uint8_t MAX_RT:1; /*Флаг прерывания достижения максимального количества повторных
передач.*/
 uint8_t TX_DS:1; /*Флаг прерывания завершения передачи данных.*/
 uint8_t RX_DR:1; /*Флаг прерывания завершения приема данных.*/
 uint8_t RESERVED:1; /*Зарезервировано.*/
};

/*Структура регистра "OBSERVE_TX".*/
struct RFIR_OBSERVE_TX_ByteField_TypeDef
```

```
{
 uint8_t ARC_CNT:4; /*Счетчик повторно переданных пакетов.*/
 uint8_t PLOS_CNT:4; /*Счетчик патеряных пакетов.*/
};

/*Структура регистра "RPD".*/
struct RFIR_RPD_ByteField_TypeDef
{
 uint8_t RPD:1; /*Бит приемника детектора мощности.*/
 uint8_t RESERVED:7; /*Зарезервировано.*/
};

/*Структура регистра "RX_PW_PX".*/
struct RFIR_RX_PW_PX_ByteField_TypeDef
{
 uint8_t RX_PW_PX:6; /*Количество принятых байт в логическом канале X.*/
 uint8_t RESERVED:2; /*Зарезервировано.*/
};

/*Структура регистра "FIFO_STATUS".*/
struct RFIR_FIFO_STATUS_ByteField_TypeDef
{
 uint8_t RX_EMPTY:1; /*Флаг опустошения буфера приемника.*/
 uint8_t RX_FULL:1; /*Флаг заполнения буфера приемника.*/
 uint8_t RESERVED1:2; /*Зарезервировано.*/
 uint8_t TX_EMPTY:1; /*Флаг опустошения буфера передатчика.*/
 uint8_t TX_FULL:1; /*Флаг заполнения буфера передатчика.*/
 uint8_t TX_REUSE:1; /*Передача последнеиспользуемой полезной нагрузки.*/
 uint8_t RESERVED2:1; /*Зарезервировано.*/
};

/*Структура регистра "DYNPD".*/
struct RFIR_DYNPD_ByteField_TypeDef
{
 uint8_t DPL_P0:1; /*Включение динамической полезной нагрузки для логического канала 0.*/
 uint8_t DPL_P1:1; /*Включение динамической полезной нагрузки для логического канала 1.*/
 uint8_t DPL_P2:1; /*Включение динамической полезной нагрузки для логического канала 2.*/
 uint8_t DPL_P3:1; /*Включение динамической полезной нагрузки для логического канала 3.*/
 uint8_t DPL_P4:1; /*Включение динамической полезной нагрузки для логического канала 4.*/
 uint8_t DPL_P5:1; /*Включение динамической полезной нагрузки для логического канала 5.*/
 uint8_t RESERVED:2; /*Зарезервировано.*/
};

/*Структура регистра "FEATURE".*/
struct RFIR_FEATURE_ByteField_TypeDef
{
 uint8_t EN_DYN_ACK:1; /*Включение команды
"RFIR_COM_TYPE_W_TX_PAYLOAD_NOACK".*/
 uint8_t EN_ACK_PAY:1; /*Включение подтверждения пакета с передачей полезной
нагрузкой.*/
 uint8_t EN_DPL:1; /*Включение динамической длины полезной нагрузки.*/
 uint8_t RESERVED:5; /*Зарезервировано.*/
```

```
};

/*Описание регистра "CONFIG".*/
union RFIR_CONFIG_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "CONFIG".*/
 struct RFIR_CONFIG_ByteField_TypeDef RFIR_CONFIG_ByteField; /*Структура регистра
"CONFIG".*/
};

/*Описание регистра "EN_AA".*/
union RFIR_EN_AA_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "EN_AA".*/
 struct RFIR_EN_AA_ByteField_TypeDef RFIR_EN_AA_ByteField; /*Структура регистра
"EN_AA".*/
};

/*Описание регистра "EN_RXADDR".*/
union RFIR_EN_RXADDR_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "EN_RXADDR".*/
 struct RFIR_EN_RXADDR_ByteField_TypeDef RFIR_EN_RXADDR_ByteField; /*Структура
регистра "EN_RXADDR".*/
};

/*Описание регистра "SETUP_AW".*/
union RFIR_SETUP_AW_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "SETUP_AW".*/
 struct RFIR_SETUP_AW_ByteField_TypeDef RFIR_SETUP_AW_ByteField; /*Структура
регистра "SETUP_AW".*/
};

/*Описание регистра "SETUP_RETR".*/
union RFIR_SETUP_RETR_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "SETUP_RETR".*/
 struct RFIR_SETUP_RETR_ByteField_TypeDef RFIR_SETUP_RETR_ByteField; /*Структура
регистра "SETUP_RETR".*/
};

/*Описание регистра "RF_CH".*/
union RFIR_RF_CH_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "RF_CH".*/
 struct RFIR_RF_CH_ByteField_TypeDef RFIR_RF_CH_ByteField; /*Структура регистра
"RF_CH".*/
};

/*Описание регистра "RF_SETUP".*/
union RFIR_RF_SETUP_Register_TypeDef
```

```
{
 uint8_t byte; /*Байт регистра "RF_SETUP".*/
 struct RFIR_RF_SETUP_ByteField_TypeDef RFIR_RF_SETUP_ByteField; /*Структура регистра
"RF_SETUP".*/
};

/*Описание регистра "STATUS".*/
union RFIR_STATUS_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "STATUS".*/
 struct RFIR_STATUS_ByteField_TypeDef RFIR_STATUS_ByteField; /*Структура регистра
"STATUS".*/
};

/*Описание регистра "OBSERVE_TX".*/
union RFIR_OBSERVE_TX_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "OBSERVE_TX".*/
 struct RFIR_OBSERVE_TX_ByteField_TypeDef RFIR_OBSERVE_TX_ByteField; /*Структура
регистра "OBSERVE_TX".*/
};

/*Описание регистра "RPD".*/
union RFIR_RPD_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "RPD".*/
 struct RFIR_RPD_ByteField_TypeDef RFIR_RPD_ByteField; /*Структура регистра "RPD".*/
};

/*Описание регистра "RX_PW_PX".*/
union RFIR_RX_PW_PX_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "RX_PW_PX".*/
 struct RFIR_RX_PW_PX_ByteField_TypeDef RFIR_RX_PW_PX_ByteField; /*Структура
регистра "RX_PW_PX".*/
};

/*Описание регистра "FIFO_STATUS".*/
union RFIR_FIFO_STATUS_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "FIFO_STATUS".*/
 struct RFIR_FIFO_STATUS_ByteField_TypeDef RFIR_FIFO_STATUS_ByteField; /*Структура
регистра "FIFO_STATUS".*/
};

/*Описание регистра "DYNPD".*/
union RFIR_DYNPD_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "DYNPD".*/
 struct RFIR_DYNPD_ByteField_TypeDef RFIR_DYNPD_ByteField; /*Структура регистра
"DYNPD".*/
};
```

```
/*Описание регистра "FEATURE".*/
union RFIR_FEATURE_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "FEATURE".*/
 struct RFIR_FEATURE_ByteField_TypeDef RFIR_FEATURE_ByteField; /*Структура регистра
"FEATURE".*/
};

/*Регистры радомодуля.*/
struct RFIR_Registers_TypeDef
{
 union RFIR_CONFIG_Register_TypeDef CONFIG_Register; /*Регистр "CONFIG".*/
 union RFIR_EN_AA_Register_TypeDef EN_AA_Register; /*Регистр "EN_AA".*/
 union RFIR_EN_RXADDR_Register_TypeDef EN_RXADDR_Register; /*Регистр
"EN_RXADDR".*/
 union RFIR_SETUP_AW_Register_TypeDef SETUP_AW_Register; /*Регистр "SETUP_AW".*/
 union RFIR_SETUP_RETR_Register_TypeDef SETUP_RETR_Register; /*Регистр
"SETUP_RETR".*/
 union RFIR_RF_CH_Register_TypeDef RF_CH_Register; /*Регистр "RF_CH".*/
 union RFIR_RF_SETUP_Register_TypeDef RF_SETUP_Register; /*Регистр "RF_SETUP".*/
 union RFIR_STATUS_Register_TypeDef STATUS_Register; /*Регистр "STATUS".*/
 union RFIR_OBSERVE_TX_Register_TypeDef OBSERVE_TX_Register; /*Регистр
"OBSERVE_TX".*/
 union RFIR_RPD_Register_TypeDef RPD_Register; /*Регистр "RPD".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P0_Register; /*Регистр
"RFIR_RX_PW_P0".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P1_Register; /*Регистр
"RFIR_RX_PW_P1".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P2_Register; /*Регистр
"RFIR_RX_PW_P2".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P3_Register; /*Регистр
"RFIR_RX_PW_P3".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P4_Register; /*Регистр
"RFIR_RX_PW_P4".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P5_Register; /*Регистр
"RFIR_RX_PW_P5".*/
 union RFIR_FIFO_STATUS_Register_TypeDef FIFO_STATUS_Register; /*Регистр
"FIFO_STATUS".*/
 union RFIR_DYNPD_Register_TypeDef DYNPD_Register; /*Регистр "DYNPD".*/
 union RFIR_FEATURE_Register_TypeDef FEATURE_Register; /*Регистр "FEATURE".*/
};

/*Основные функции для работы с модулем.*/
uint8_t RFIR_Read_Register(const uint8_t); /*Функция чтения из регистра.*/
void RFIR_Write_Register(const uint8_t,const uint8_t); /*Функция записи в регистр.*/
void RFIR_Read_Buffer(const uint8_t,uint8_t*,const uint8_t); /*Функция чтения массива данных
из радиомодуля.*/
void RFIR_Write_Buffer(const uint8_t,const uint8_t*,const uint8_t);      /*Функция записи массива
данных в радиомодуль. */

/*Основные комманды для работы с модулем.*/
```

```
uint8_t RFIR_Get_Status(void);
	/*Функция считывания регистра состояния радиомодуля.*/
void RFIR_Flush_Buffer(const enum RFIR_Command_TypeDef);	/*Функция
очистки буферов приема и передачи радиомодуля.*/
void RFIR_Run_Write(const uint8_t);
	/*Функция запуска передачи данных.*/

#endif /* RFINTERFACERELLIB_H_ */
```

Листинг файла RTOSHALLib.c

```
/*
 * Назначение: Файл модуля абстракции ОСРВ от аппаратуры.
 * Имя файла: RTOSHALLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 17.10.2014 г.
 * Список изменений:
 * 17.01.2014 г. − Исходная версия.
*/

#include "RTOSHALLib.h"

/*****************************************************************************
************
 * Прототип функции: void RTOSHALTaskTimerInitialization(const uint16_t,const uint16_t);
 * Назначение: Функция настройки системного таймера задач для МК.
 * Аргументы: "psc" − значение предделителя таймера. "arr" − значение счетчика таймера при
 * достижении которого генерируется прерывание.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки системного таймера задач для МК.

*****************************************************************************
***********/
void RTOSHALTaskTimerInitialization(const uint16_t psc,const uint16_t arr)
{
	if(psc>0)
	{
		BREAK;	/*Включить предделитель.*/
	}
	OCR1A=arr;
	SET_BIT(TIMSK,OCIE1A);
	TCCR1B=(1<<WGM12)|(1<<CS10);
}
```

Листинг заголовочного файла RTOSHALLib.h

```
/*
 * Назначение: Заголовочный файл модуля абстракции ОСРВ от аппаратуры.
 * Имя файла: RTOSHALLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 17.10.2014 г.
 * Список изменений:
 * 17.01.2014 г. − Исходная версия.
*/

#ifndef RTOSHALLIB_H_
#define RTOSHALLIB_H_

#include "GenDefineLib.h"

#define RTOS_MAX_TIME_COUNTER_VALUE CONFIG_MAX_TIME_COUNTER_VALUE
        /*Максимальное число помещающееся в регистре таймера.*/
#define RTOS_TIME_QUANT CONFIG_TIME_QUANT
                /*Квант времени планировщика в с.*/
#define RTOS_MCU_FREQUENCY CONFIG_MCU_FREQUENCY
                /*Тактовая частота таймера в Гц.*/

void RTOSHALTaskTimerInitialization(const uint16_t,const uint16_t);          /*Функция
настройки системного таймера задач для МК.*/

#endif /* RTOSHALLIB_H_ */
```

Листинг файла RTOSShedulerLib.c

```
/*
 * Назначение: Файл модуля планировщика задач ОСРВ.
 * Имя файла: RTOSShedulerLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 17.01.2014 г.
 * Список изменений:
 * 17.01.2014 г. − Исходная версия.
 * 24.03.2014 г. − Исполняемые задачи описываются массивом структур.
 * 06.06.2014 г. − Структуры планировщика и исполняемые задачи внесены в модуль планировщика.
 * 17.10.2014 г. − Изменена структура планировщика.
*/

#include "RTOSShedulerLib.h"

struct RTOSTask_TypeDef RTOSSystemTask[]=
{
	{
		RTOS_TIME_TASK_0,
		RTOS_TIME_TASK_0,
		RTOS_TASK_STATUS_REDY,
	},
	{
		RTOS_TIME_TASK_1,
		RTOS_TIME_TASK_1,
		RTOS_TASK_STATUS_REDY,
	},
};

struct RTOS_ADC_Data_TypeDef RTOS_ADC_Data=
{
	RTOS_DATA_STATUS_NULL,
	0,
	ADC_RESISTOR_VALUE_0_00,
};

/*Блок определений внутренних функций.*/
void RTOSInitialization(void); /*Функция настройки временного кванта планировщика.*/
void RTOSTaskFunction0(void); /*Функция выполнения задачи 0.*/
void RTOSTaskFunction1(void); /*Функция выполнения задачи 1.*/

const enum ADCReistor_TypeDef CurrentResist = ADC_RESISTOR_VALUE_5_46; /*Текущее значение
сопротивления.*/

void SendDataToHost(const uint16_t, const enum ADCReistor_TypeDef);	/*Функция генерация пакета данных и
его отправки на хост.*/

/*****************************************************************************************
 * Прототип функции: void RTOSInitialization(void);
 * Назначение: Функция настройки временного кванта планировщика.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция задает минимальное квантование времени для планировщика задач.
 *****************************************************************************************/
void RTOSInitialization(void)
{
 register unsigned int tmp=RTOS_TIME_QUANT*RTOS_MCU_FREQUENCY;
 register unsigned int i=0;
 if(tmp>RTOS_MAX_TIME_COUNTER_VALUE)
 {
 unsigned int j=tmp;
 for(i=2;tmp>RTOS_MAX_TIME_COUNTER_VALUE;i++)
```

```
 {
 if((tmp/=i)>RTOS_MAX_TIME_COUNTER_VALUE)
 {
 tmp=j;
 }
 }
 }
 RTOSHALTaskTimerInitialization((uint16_t)i,(uint16_t)tmp);
}

/*****************************************************************************************
 * Прототип функции: void RTOSTaskCounter(void);
 * Назначение: Функция инкремента счетчика задач.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция инкремента счетчика временных квантов оставшихся до исполнения задачи.
 *****************************************************************************************/
void RTOSTaskCounter(void)
{
 struct RTOSTask_TypeDef* first_element=&RTOSSystemTask[0];
 struct RTOSTask_TypeDef*
last_element=&RTOSSystemTask[(sizeof(RTOSSystemTask)/sizeof(RTOSSystemTask[0]))];
 while(first_element<last_element)
 {
 if(first_element->Status==RTOS_TASK_STATUS_REDY)
 {
 if((--first_element->TaskTimeCounter)==0)
 {
 first_element->TaskTimeCounter=first_element->TaskTime;
 first_element->Status=RTOS_TASK_STATUS_EXECUTE;
 }
 }
 first_element++;
 }
}

/*****************************************************************************************
 * Прототип функции: void RTOSTaskHandler(void);
 * Назначение: Функция запуска обработчиков задач.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция запускает и останавливает обработчики задач. Должна распологаться в
 * основном цикле main.
 *****************************************************************************************/
void RTOSTaskHandler(void)
{
 struct RTOSTask_TypeDef* first_element=&RTOSSystemTask[0];
 struct RTOSTask_TypeDef*
last_element=&RTOSSystemTask[(sizeof(RTOSSystemTask)/sizeof(RTOSSystemTask[0]))];
 register unsigned int i=0;
 while(first_element<last_element)
 {
 if(first_element->Status==RTOS_TASK_STATUS_EXECUTE)
 {
 switch(i)
 {
                    case 0:
 {
                              RTOSTaskFunction0();
                              break;
 }
                    case 1:
                    {
```

```
                        RTOSTaskFunction1();
                        break;
                }
                default:
                {
                        break;
                }
 }
 first_element->Status=RTOS_TASK_STATUS_REDY;
 }
 i++;
 first_element++;
 }
 /*Блок постоянного выполнения.*/
 {
        if (RTOS_ADC_Data.Status==RTOS_DATA_STATUS_NEW)
        {
                SendDataToHost(RTOS_ADC_Data.Data, RTOS_ADC_Data.Resist);
                RTOS_ADC_Data.Status=RTOS_DATA_STATUS_OLD;
        }
 }
}

/*****************************************************************************************
 * Прототип функции: void RTOSHardwareInitialization(void);
 * Назначение: Функция настройки аппаратуры.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки аппаратуры.
 *****************************************************************************************/
void RTOSHardwareInitialization(void)
{
        RTOSInitialization();
 RFI_Initialization();
        ADCInitialization();
        HIDIInitialization();
        sei();
        HIDILEDSetState(GENERAL_STATE_ENABLE);
}

/*****************************************************************************************
 * Прототип функции: void RTOSIRQHandler(enum RTOSInterrupt_TypeDef);
 * Назначение: Функция вызова обработчиков прерываний от аппаратуры.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция вызова обработчиков прерываний от аппаратуры.
 *****************************************************************************************/
void RTOSIRQHandler(enum RTOSInterrupt_TypeDef RTOSInterrupt)
{
        switch(RTOSInterrupt)
        {
                case RTOS_IRQ_HW_RF_MODULE:
                {
                        RFI_IRQ_Handler();
                        break;
                }
                case RTOS_IRQ_HW_ADC:
                {
                        RTOS_ADC_Data.Data=ADCHandler();
                        RTOS_ADC_Data.Resist=CurrentResist;
                        RTOS_ADC_Data.Status=RTOS_DATA_STATUS_NEW;
                        break;
                }
```

```
		default:
		{
			break;
		}
	}
}

/*****************************************************************************************
* Прототип: void RTOSTaskFunction0(void);
* Назначение: Функция выполнения задачи 0.
* Аргументы: Нет.
* Возвращаемое значение: Нет.
* Описание: Функция выполнения задачи 0.
*****************************************************************************************/
void RTOSTaskFunction0(void)
{
	if (HIDILEDGetState()==GENERAL_STATE_ENABLE)
	{
		HIDILEDSetState(GENERAL_STATE_DISABLE);
	}
}

/*****************************************************************************************
* Прототип: void RTOSTaskFunction1(void);
* Назначение: Функция выполнения задачи 1.
* Аргументы: Нет.
* Возвращаемое значение: Нет.
* Описание: Функция выполнения задачи 1.
*****************************************************************************************/
void RTOSTaskFunction1(void)
{
	if (RTOS_ADC_Data.Status!=RTOS_DATA_STATUS_NEW)
	{
		ADCSampleStart(CurrentResist);
	}
}

/*****************************************************************************************
* Прототип: void SendDataToHost(const uint16_t, const enum ADCReistor_TypeDef);
* Назначение: Функция генерация пакета данных и его отправки на хост.
* Аргументы: "data" − данные для передачи. "resist" − данные о сопротивлении для передачи.
* Возвращаемое значение: Нет.
* Описание: Функция генерация пакета данных и его отправки на хост.
*****************************************************************************************/
void SendDataToHost(const uint16_t data, const enum ADCReistor_TypeDef resist)
{
	uint8_t buffer[8];
	static uint8_t PackageCounter = 0;
	buffer[0]=START_BYTE;
	buffer[1]=GetPreambule();
	Uint16_tToByte(&buffer[2],data);
	buffer[4]=(uint8_t)resist;
	buffer[5]=PackageCounter++;
	buffer[6]=GetXor(&buffer[2],&buffer[5]);
	buffer[7]=STOP_BYTE;
	RFI_Send_Buffer(&buffer[0],MASSIV_SIZE(buffer));
}
```

Листинг заголовочного файла RTOSShedulerLib.h

```
/*
 * Назначение: Заголовочный файл модуля планировщика задач ОСРВ.
 * Имя файла: RTOSShedulerLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 17.01.2014 г.
 * Список изменений:
 * 17.01.2014 г. − Исходная версия.
 * 24.03.2014 г. − Исполняемые задачи описываются массивом структур.
 * 06.06.2014 г. − Структуры планировщика и исполняемые задачи внесены в модуль планировщика.
 * 17.10.2014 г. − Изменена структура планировщика.
 */

#ifndef RTOSSHEDULERLIB_H_
#define RTOSSHEDULERLIB_H_

#include "RTOSHALLib.h"
#include "RFInterfaceLib.h"
#include "ADCInterfaceLib.h"
#include "HIDInterfaceLib.h"

#define RTOS_TIME_TASK_0     (CONFIG_LED_BLINK/CONFIG_TIME_QUANT)            /*Период исполнения
задачи 0, исчисляется в квантах времени планировщика.*/
#define RTOS_TIME_TASK_1     ((1.0/CONFIG_START_ADC_SEMPLE)/CONFIG_TIME_QUANT) /*Период
исполнения задачи 1, исчисляется в квантах времени планировщика.*/

/*Режимы состояния выполнения задачи.*/
enum RTOSTaskStatus_TypeDef
{
 RTOS_TASK_STATUS_REDY, /*Выполнение задачи запрещено.*/
 RTOS_TASK_STATUS_EXECUTE, /*Выполнение задачи разрешено.*/
};

/*Прерывания с которым работает ОСРВ.*/
enum RTOSInterrupt_TypeDef
{
	RTOS_IRQ_HW_ADC,		/*Прерывание от АЦП.*/
	RTOS_IRQ_HW_RF_MODULE,	/*Прерывание от радиомодуля.*/
	RTOS_IRQ_HW_NULL,		/*Отсутствие прерываний от аппаратуры.*/
	RTOS_IRQ_SW_NULL,		/*Отсутствие прерываний от программных модулей.*/
};

/*Состояние данных в буфере.*/
enum RTOSDataStatus_TypeDef
{
 RTOS_DATA_STATUS_OLD, /*Данные не актуальны.*/
 RTOS_DATA_STATUS_NEW, /*Данные актуальны.*/
 RTOS_DATA_STATUS_ERROR, /*В данных ошибка.*/
 RTOS_DATA_STATUS_NULL, /*Данные не приняты.*/
};

/*Структура описания задачи.*/
struct RTOSTask_TypeDef
{
 const unsigned int TaskTime; /*Период исполнениязадачи в минимальных квантах времени ОСРВ.*/
 unsigned int TaskTimeCounter; /*Счетчик временных квантов оставшихся до исполнения задачи.*/
 enum RTOSTaskStatus_TypeDef Status; /*Статус состояния выполнения задачи.*/
};

/*Данные с АЦП.*/
struct RTOS_ADC_Data_TypeDef
{
```

```
	enum RTOSDataStatus_TypeDef Status; /*Состояние данных в буфере.*/
	uint16_t Data;	/*Данные.*/
	enum ADCReistor_TypeDef Resist;	/*Сопротивление.*/
};

void RTOSHardwareInitialization(void); /*Функция настройки аппаратуры.*/
void RTOSIRQHandler(enum RTOSInterrupt_TypeDef); /*Функция вызова обработчиков прерываний от аппаратуры.*/
void RTOSTaskCounter(void); /*Функция инкремента счетчика задач.*/
void RTOSTaskHandler(void); /*Функция запуска обработчиков задач.*/

#endif /* RTOSSHEDULERLIB_H_ */
```

ЛИСТИНГ МИКРОПРОГРАММЫ УСТРОЙСТВА СОПРЯЖЕНИЯ С ПЭВМ

Листинг файла Config.h

```
/*
 * Назначение: Конфигурационный файл.
 * Имя файла: Config.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 18.04.2015 г.
 * Список изменений:
 * 18.04.2015 г. − Исходная версия.
 */

#ifndef CONFIG_H_
#define CONFIG_H_

#define F_CPU 8000000UL

/*Настройка OCPB*/
#define CONFIG_MAX_TIME_COUNTER_VALUE 65535        /*Максимальное число
помещающееся в регистре таймера.*/
#define CONFIG_TIME_QUANT 0.002500                 /*Квант времени планировщика в
с.*/
#define CONFIG_MCU_FREQUENCY F_CPU                 /*Тактовая частота таймера в
Гц.*/

/*Настройка задач*/
#define CONFIG_LED_BLINK 1.0 /*Отправка пакета данных (сек.).*/

/*Настройка USART*/
#define CONFIG_UART_BUFFER_LENGTH 8    /*Максимальная длина буфера данных.*/
#define CONFIG_UART_BAUDRATE 9600L     /*Скорость передачи в бод.*/

#endif /* CONFIG_H_ */
```

Листинг файла GenDefineLib.h

```
/*
 * Назначение: Определение системных макросов, констант и определений.
 * Имя файла: GenDefineLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 16.10.2014 г.
 * Список изменений:
 * 16.10.2014 г. − Исходная версия.
 * 28.11.2014 г. − Добавлены определения байтов преамбулы и инструкция __nop().
 * 21.04.2015 г. − Добавлены макросы MODIFY_REG, SET_BITS, CLEAR_BITS.
 */

#ifndef GENDEFINELIB_H_
#define GENDEFINELIB_H_

#include "Config.h"
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define SET_BIT(REG,BIT)          (REG|=(1<<BIT))
#define CLEAR_BIT(REG,BIT)        (REG&=~(1<<BIT))
#define SET_BITS(REG,BIT)         (REG|=(BIT))
#define CLEAR_BITS(REG,BIT)       (REG&=~(BIT))
#define WRITE_REG(REG,BIT)        (REG=(BIT))
#define READ_REG(REG) (REG)
#define MODIFY_REG(REG,CLEARMASK,SETMASK)
        WRITE_REG((REG),(((READ_REG(REG))&(~(CLEARMASK)))|(SETMASK)))
#define TRUE (1==1)
#define FALSE (0==1)
#define SET TRUE
#define CLEAR FALSE
#define CHECKED_BIT(REG,BIT) (((REG)&(1<<BIT))?TRUE:FALSE)
#define HIGH(REG) ((REG)>>8)
#define LOW(REG) ((REG)&255)
#define BREAK while(1) /*Используется при отладке.*/
#define START_BYTE 0xAA /*Значение "Старт−байта в пакете.*/
#define STOP_BYTE 0x55 /*Значение "Стоп−байта в пакете.*/
#define PREAMBULE_BYTE_1 0xA5                    /*Байт преамбулы 1.*/
#define PREAMBULE_BYTE_2 0x5A                    /*Байт преамбулы 2.*/

#define __nop() asm volatile ("NOP")        /*Вызов инструкции "NOP".*/

/*Общие состояния.*/
enum GeneralState_TypeDef
{
 GENERAL_STATE_TRUE, /*"Истина".*/
 GENERAL_STATE_FALSE, /*"Ложь".*/
 GENERAL_STATE_SET, /*"Установлен".*/
 GENERAL_STATE_CLEAR, /*"Снят".*/
 GENERAL_STATE_ENABLE, /*"Включено".*/
 GENERAL_STATE_DISABLE, /*"Выключено".*/
 GENERAL_STATE_NULL, /*Состояние не определено.*/
};

#endif /* GENDEFINELIB_H_ */
```

Листинг файла GeneralAlgorithmsLib.c

```
/*
 * Назначение: Файл модуля основных алгоритмов микропрограммы.
 * Имя файла: GeneralAlgorithmsLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 30.10.2014
 * Список изменений: 30.10.2014 − Исходная версия.
 */

#include "GeneralAlgorithmsLib.h"

/*****************************************************************************************
 * Прототип функции: float GAAverSumFunc(struct GAAverageBuffer_TypeDef*,const float);
 * Назначение: Функция вычисляет среднее значение суммы чисел поступающих на ее вход методом \
 * скользящего окна.
 * Аргументы: "buffer" − буфер усредняемых значений. "number" − добавляемое к устреднению число.
 * Возвращаемые значения: Вычисленная средняя величина.
 * Описание: Функция вычисляет среднее значение суммы чисел поступающих на ее вход методом \
 * скользящего окна.
 *****************************************************************************************/
float GAAverSumFunc(struct GAAverageBuffer_TypeDef* buffer,const float number)
{
	buffer−>Sum−=buffer−>buffer[buffer−>pBuffer];
	buffer−>buffer[buffer−>pBuffer++]=number;
	buffer−>Sum+=number;
	if(buffer−>pBuffer==buffer−>BufferLength)
	{
		buffer−>pBuffer=0;
	}
	return buffer−>Sum/buffer−>BufferLength;
}

/*****************************************************************************************
 * Прототип функции: uint8_t GetXor(const uint8_t*,const uint8_t*);
 * Назначение: Функция получения байта контрольной суммы массива байт.
 * Аргументы: "firstElement" − первый элемент массива. "lastElement" − последний элемент \
 * массива.
 * Возвращаемые значения: Контрольная сумма массива байт типа "uint8_t".
 * Описание: Функция вычисляет контрольную сумму массива байт методом "исключающего ИЛИ".
 *****************************************************************************************/
uint8_t GetXor(const uint8_t* firstElement,const uint8_t* lastElement)
{
	register uint8_t tmp=0;
	if(firstElement!=lastElement)
	{
		tmp=*firstElement++;
		while(firstElement!=lastElement+1)
		{
			tmp^=*firstElement++;
		}
	}
 return tmp;
}

/*****************************************************************************************
 * Прототип функции: uint16_t ByteToUint16_t(const uint8_t*);
 * Назначение: Функция получения числа типа "uint16_t" из массива байт.
 * Аргументы: "buffer" − указатель на первый элемент числа в буфере.
 * Возвращаемые значения: Число типа "uint16_t".
 * Описание: Функция получения числа типа "uint16_t" из массива байт.
 *****************************************************************************************/
uint16_t ByteToUint16_t(const uint8_t* buffer)
```

```
{
	union Uint16_T_Convert_TypeDef tmp;
	tmp.byte[0]=buffer[0];
	tmp.byte[1]=buffer[1];
	return tmp.data;
}

/*****************************************************************************************
 * Прототип функции: void Uint16_tToByte(uint8_t*,const uint16_t);
 * Назначение: Функция получения массива байт из числа типа "uint16_t".
 * Аргументы: "buffer" − указатель на первый элемент числа в буфере. "value" − преобразовываемое число.
 * Возвращаемые значения: Нет.
 * Описание: Функция получения массива байт из числа типа "uint16_t".
 *****************************************************************************************/
void Uint16_tToByte(uint8_t* buffer,const uint16_t value)
{
	union Uint16_T_Convert_TypeDef tmp;
	tmp.data=value;
	buffer[0]=tmp.byte[0];
	buffer[1]=tmp.byte[1];
}
```

Листинг заголовочный файла GeneralAlgorithmsLib.h

```
/*
 * Назначение: Заголовочный файл модуля основных алгоритмов микропрограммы.
 * Имя файла: GeneralAlgorithmsLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 30.10.2014
 * Список изменений: 30.10.2014 − Исходная версия.
 */

#ifndef GENERALALGORITHMSLIB_H_
#define GENERALALGORITHMSLIB_H_

#include "GenDefineLib.h"

/*Буфер для усреднения значений методом скользящего окна.*/
struct GAAverageBuffer_TypeDef
{
 float* buffer; /*Указатель на массив буфера.*/
 const unsigned int BufferLength; /*Длина буфера.*/
 float Sum; /*Сумма элементов буфера.*/
 unsigned int pBuffer; /*Указатель на элемент буфера.*/
};

/*Объединение для преобразования полуслова.*/
union Uint16_T_Convert_TypeDef
{
	uint16_t data;				/*Слово.*/
	uint8_t byte[sizeof(uint16_t)];		/*Массив байт.*/
};

float GAAverSumFunc(struct GAAverageBuffer_TypeDef*,const float); /*Функция вычисляет
среднее значение суммы чисел поступающих на ее вход методом скользящего окна.*/
uint8_t GetXor(const uint8_t*,const uint8_t*); /*Функция получения байта контрольной суммы
массива байт.*/
uint16_t ByteToUint16_t(const uint8_t*);	/*Функция
получения числа типа "uint16_t" из массива байт.*/
void Uint16_tToByte(uint8_t*,const uint16_t);	/*Функция
получения массива байт из числа типа "uint16_t".*/

#endif /* GENERALALGORITHMSLIB_H_ */
```

Листинг заголовочный файла HIDInterfaceHALLib.h

```
/*
 * Назначение: Заголовочный файл модуля абстракции интерфейса взаимодействия с
пользователем.
 * Имя файла: HIDInterfaceHALLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 29.10.2014 г.
 * Список изменений:
 * 29.10.2014 г. − Исходная версия.
 */

#ifndef HIDINTERFACEHALLIB_H_
#define HIDINTERFACEHALLIB_H_

#include "GenDefineLib.h"

#define HIDIHLEDInit()      SET_BIT(DDRD,PD5)           /*Функция инициализации порта
светодиода.*/
#define HIDIHLEDSet() CLEAR_BIT(PORTD,PD5)      /*Включить светодиод.*/
#define HIDIHLEDClear()   SET_BIT(PORTD,PD5)            /*Погасить светодиод.*/

#endif /* HIDINTERFACEHALLIB_H_ */
```

Листинг файла HIDInterfaceLib.c

```
/*
 * Назначение: Файл модуля интерфейса взаимодействия с пользователем.
 * Имя файла: HIDInterfaceLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 29.10.2014 г.
 * Список изменений:
 * 29.10.2014 г. − Исходная версия.
 */

#include "HIDInterfaceLib.h"

enum GeneralState_TypeDef LEDstate=GENERAL_STATE_DISABLE; /*Начальное состояние светодиода.*/

/*****************************************************************************************
 * Прототип функции: void HIDIInitialization(void):
 * Назначение: Функция инициализации интерфейса с пользователем.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция инициализации интерфейса с пользователем.
 *****************************************************************************************/
void HIDIInitialization(void)
{
 HIDIHLEDInit();
 HIDILEDSetState(LEDstate);
}

/*****************************************************************************************
 * Прототип функции: void HIDILEDSetState(enum GeneralState_TypeDef);
 * Назначение: Функция задания режима работы светодиода "Включено"/"Выключено".
 * Аргументы: "state" − устанавливаемое состояние светодиодов.
 * Возвращаемые значения: Нет.
 * Описание: Функция задания режима работы светодиода "Включено"/"Выключено".
 *****************************************************************************************/
void HIDILEDSetState(enum GeneralState_TypeDef state)
{
 if(state==GENERAL_STATE_ENABLE)
 {
 LEDstate=GENERAL_STATE_ENABLE;
 HIDIHLEDSet();
 }
 else
 {
 LEDstate=GENERAL_STATE_DISABLE;
 HIDIHLEDClear();
 }
}

/*****************************************************************************************
 * Прототип функции: void HIDILEDChangeState(void);
 * Назначение: Функция изменения состояния светодиода на противоположенное.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция изменения состояния светодиода на противоположенное.
 *****************************************************************************************/
void HIDILEDChangeState(void)
{
 if(LEDstate!=GENERAL_STATE_ENABLE)
 {
 HIDILEDSetState(GENERAL_STATE_ENABLE);
 }
 else
```

```
 {
 HIDILEDSetState(GENERAL_STATE_DISABLE);
 }
}

/*****************************************************************************************
 * Прототип функции: enum GeneralState_TypeDef HIDILEDGetState(void);
 * Назначение: Функция получения состояния светодиода.
 * Аргументы: Нет.
 * Возвращаемые значения: Состояние светодиода.
 * Описание: Функция получения состояния светодиода.
 *****************************************************************************************/
enum GeneralState_TypeDef HIDILEDGetState(void)
{
 return LEDstate;
}
```

Листинг заголовочного файла HIDInterfaceLib.h

```
/*
 * Назначение: Заголовочный файл модуля интерфейса взаимодействия с пользователем.
 * Имя файла: HIDInterfaceLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 29.10.2014 г.
 * Список изменений:
 * 29.10.2014 г. − Исходная версия.
 */

#ifndef HIDINTERFACELIB_H_
#define HIDINTERFACELIB_H_

#include "HIDInterfaceHALLib.h"

void HIDIInitialization(void); /*Функция инициализации интерфейса с пользователем.*/
void HIDILEDSetState(enum GeneralState_TypeDef); /*Функция задания режима работы
светодиода "Включено"/"Выключено".*/
void HIDILEDChangeState(void); /*Функция изменения состояния светодиода на
противоположенное.*/
enum GeneralState_TypeDef HIDILEDGetState(void); /*Функция получения состояния
светодиода.*/

#endif /* HIDINTERFACELIB_H_ */
```

Листинг файла CaudaScopeHost.c

```
/*
 * Назначение: Основной файл микропрограммы.
 * Имя файла: CaudaScopeHost.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 18.04.2015 г.
 * Список изменений: 18.04.2015 г. − Исходная версия.
 */

#include "GenDefineLib.h"
#include "RTOSShedulerLib.h"

ISR(TIMER1_COMPA_vect)
{
	RTOSTaskCounter();
}

ISR(INT0_vect)
{
	RTOSIRQHandler(RTOS_IRQ_HW_RF_MODULE);
}

ISR(USART_UDRE_vect)
{
	RTOSIRQHandler(RTOS_IRQ_HW_UART);
}

/*****************************************************************************
************
 * Прототип функции: int main(void)
 * Назначение: Главная функция микропрограммы.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Главная функция микропрограммы.

*****************************************************************************
***********/
int main(void)
{
	RTOSHardwareInitialization();
	while(1)
	{
		RTOSTaskHandler();
	}
}
```

Листинг файла RFInterfaceHALLib.c

```
/*
 * Назначение: Файл модуля абстракции радиоинтерфейса от аппаратуры.
 * Имя файла: RFInterfaceHALLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 23.10.2014 г.
 * Список изменений:
 * 23.10.2014 г. − Исходная версия.
 */

#include "RFInterfaceHALLib.h"

/*Блок объявления внутренних функций.*/
void RFIH_Port_Init(void); /*Функция настройки портов для взаимодействия с радиомодулем.*/
void RFIH_IRQ_Init(void); /*Функция настройки запроса прерывания от радиомодуля.*/
void RFIH_SPI_Init(void); /*Функция настройки интерфейса SPI.*/

/*****************************************************************************************
 * Прототип функции: void RFIH_Hardware_Init(void);
 * Назначение: Функция настройки аппаратного обеспечения для взаимодействия с радиомодулем.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки аппаратного обеспечения для взаимодействия с радиомодулем.
 *****************************************************************************************/
void RFIH_Hardware_Init(void)
{
 RFIH_Port_Init();
 RFIH_SPI_Init();
 RFIH_IRQ_Init();
}

/*****************************************************************************************
 * Прототип функции: void RFIH_Port_Init(void);
 * Назначение: Функция настройки портов для взаимодействия с радиомодулем.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки портов для взаимодействия с радиомодулем.
 *****************************************************************************************/
void RFIH_Port_Init(void)
{
	MODIFY_REG(RF_PORT,(1<<VCC),(1<<CE));
	SET_BITS(RF_DDR,((1<<VCC)|(1<<CE)));
	SET_BITS(SPI_PORT,((1<<SS)|(1<<MISO)|(1<<MOSI)|(1<<SCK)));
	SET_BITS(SPI_DDR,((1<<SS)|(1<<MOSI)|(1<<SCK)|(1<<MISO)));
}

/*****************************************************************************************
 * Прототип функции: void RFIH_IRQ_Init(void);
 * Назначение: Функция настройки запроса прерывания от радиомодуля.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки запроса прерывания от радиомодуля.
 *****************************************************************************************/
void RFIH_IRQ_Init(void)
{
	CLEAR_BITS(MCUCR,((1<<ISC01)|(1<<ISC00)));
	SET_BIT(GICR,INT0);
}

/*****************************************************************************************
 * Прототип функции: void RFIH_SPI_Init(void);
 * Назначение: Функция настройки интерфейса SPI.
```

```
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настраивает интерфейс SPI в режиме "Ведущего".
 ***********************************************************************************************/
void RFIH_SPI_Init(void)
{
	MODIFY_REG(SPCR,((1<<SPIE)|(1<<DORD)|(1<<CPOL)|(1<<CPHA)|(1<<SPR1)|(1<<SPR0)),((1<<SPE)|(1<
<MSTR)));
	SET_BIT(SPSR,SPI2X);
}

/***********************************************************************************************
 * Прототип функции: void RFIH_Powered(void);
 * Назначение: Функция включения подачи питания на радиомодуль.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция включения подачи питания на радиомодуль.
 ***********************************************************************************************/
void RFIH_Powered(void)
{
	SET_BIT(RF_PORT,VCC);
	_delay_ms(120);
}

/***********************************************************************************************
 * Прототип функции: void RFIH_Unpowered(void);
 * Назначение: Функция выключения подачи питания на радиомодуль.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция выключения подачи питания на радиомодуль.
 ***********************************************************************************************/
void RFIH_Unpowered(void)
{
	CLEAR_BIT(RF_PORT,VCC);
	_delay_ms(120);
}
```

Листинг заголовочный файла RFInterfaceHALLib.h

```
/*
 * Назначение: Заголовочный файл модуля абстракции радиоинтерфейса от аппаратуры.
 * Имя файла: RFInterfaceHALLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 23.10.2014 г.
 * Список изменений:
 * 23.10.2014 г. − Исходная версия.
 */

#ifndef RFINTERFACEHALLIB_H_
#define RFINTERFACEHALLIB_H_

#include "GenDefineLib.h"

/*номера потров радиоинтерфейса.*/
#define VCC	0
#define CE	1

/*Номера портов интерфейса SPI.*/
#define SS		4
#define MOSI	5
#define MISO	6
#define SCK		7

#define SPI_DDR DDRB /*Порт интерфейса SPI.*/
#define SPI_PORT PORTB		/*Порт интерфейса SPI.*/

#define RF_PORT PORTB		/*Выводы радиоинтерфейса.*/
#define RF_DDR DDRB		/*Порт радиоинтерфейса.*/

#define RFIH_CSN_Pin_Clear()		CLEAR_BIT(RF_PORT,SS)		/*Установить
низкий логический уровень на выводе CSN.*/
#define RFIH_CSN_Pin_Set()		SET_BIT(RF_PORT,SS)
	/*Установить высокий логический уровень на выводе CSN.*/
#define RFIH_CE_Pin_Set()		SET_BIT(RF_PORT,CE)
	/*Установить высокий логический уровень на выводе CE.*/
#define RFIH_CE_Pin_Clear()		CLEAR_BIT(RF_PORT,CE)
	/*Установить низкий логический уровень на выводе CE.*/
#define RFIH_Write_Byte(byte)		WRITE_REG(SPDR,(uint8_t)byte)		/*Записать байт.*/
#define RFIH_Read_Byte()		(uint8_t)SPDR
	/*Считать байт.*/
#define RFIH_Check_End_Transaction()	CHECKED_BIT(SPSR,SPIF)		/*Проверить завершение
передачи.*/
#define RFIH_IRQ_Acknowledgement()		SET_BIT(GIFR,INTF0)
	/*Подтверждение обработки запроса прерывания.*/

void RFIH_Hardware_Init(void); /*Функция настройки аппаратного обеспечения для взаимодействия с
радиомодулем.*/
void RFIH_Powered(void);		/*Функция включения подачи питания на радиомодуль.*/
void RFIH_Unpowered(void);		/*Функция выключения подачи питания на радиомодуль.*/

#endif /* RFINTERFACEHALLIB_H_ */
```

Листинг файла RFInterfaceLib.c

```
/*
 * Назначение: Файл модуля радиоинтерфейса.
 * Имя файла: RFInterfaceLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 23.10.2014 г.
 * Список изменений:
 * 23.10.2014 г. − Исходная версия.
 */

#include "RFInterfaceLib.h"

uint8_t RX_Buffer[8];                                                    /*Буфер
принемаемых данных.*/
uint8_t RFI_Device_Address[]={0x40,0x50,0x60,0x70,0x80};        /*Адрес устройства.*/

struct RFIR_Registers_TypeDef RF_Device_Register;                /*Регистры устройства.*/

/*****************************************************************************************
 * Прототип функции: void RFI_Initialization(void);
 * Назначение: Функция настройки радиомодуля.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки радиомодуля.
 *****************************************************************************************/
void RFI_Initialization(void)
{
	RFIH_Hardware_Init();
	RFIH_CE_Pin_Clear();
	RFIH_Unpowered();
	RFIH_Powered();
	RFI_Read_Device_Registers();
	/*Общие настройки.*/

	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P0=CLEAR;
	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P1=CLEAR;
	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P2=CLEAR;
	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P3=CLEAR;
	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P4=CLEAR;
	RF_Device_Register.EN_AA_Register.RFIR_EN_AA_ByteField.ENAA_P5=CLEAR;
	RFIR_Write_Register(RFIR_REG_TYPE_EN_AA,RF_Device_Register.EN_AA_Register.byte);

	RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P0=SET;
	RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P1=CLEAR;
	RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P2=CLEAR;
	RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P3=CLEAR;
	RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P4=CLEAR;
	RF_Device_Register.EN_RXADDR_Register.RFIR_EN_RXADDR_ByteField.ERX_P5=CLEAR;
	RFIR_Write_Register(RFIR_REG_TYPE_EN_RXADDR,RF_Device_Register.EN_RXADDR_Register.byte);

	RF_Device_Register.SETUP_AW_Register.RFIR_SETUP_AW_ByteField.AW=(5−2);
	RFIR_Write_Register(RFIR_REG_TYPE_SETUP_AW,RF_Device_Register.SETUP_AW_Register.byte);

	RF_Device_Register.SETUP_RETR_Register.RFIR_SETUP_RETR_ByteField.ARD=(750/250−1);
	RF_Device_Register.SETUP_RETR_Register.RFIR_SETUP_RETR_ByteField.ARC=10;
	RFIR_Write_Register(RFIR_REG_TYPE_SETUP_RETR,RF_Device_Register.SETUP_RETR_Register.byte);

	RF_Device_Register.RF_CH_Register.RFIR_RF_CH_ByteField.RF_CH=40;
	RFIR_Write_Register(RFIR_REG_TYPE_RF_CH,RF_Device_Register.RF_CH_Register.byte);

	RF_Device_Register.RF_SETUP_Register.RFIR_RF_SETUP_ByteField.RF_DR_HIGH=CLEAR;
	RF_Device_Register.RF_SETUP_Register.RFIR_RF_SETUP_ByteField.RF_DR_LOW=SET;
```

```
	RF_Device_Register.RF_SETUP_Register.RFIR_RF_SETUP_ByteField.RF_PWR=3;
	RFIR_Write_Register(RFIR_REG_TYPE_RF_SETUP,RF_Device_Register.RF_SETUP_Register.byte);

	RF_Device_Register.DYNPD_Register.byte=0;
	RFIR_Write_Register(RFIR_REG_TYPE_DYNPD,RF_Device_Register.DYNPD_Register.byte);

	RF_Device_Register.FEATURE_Register.byte=0;
	RFIR_Write_Register(RFIR_REG_TYPE_FEATURE,RF_Device_Register.FEATURE_Register.byte);

	RFIR_Write_Buffer(RFIR_REG_TYPE_RX_ADDR_P0,RFI_Device_Address,sizeof(RFI_Device_Address)/size
of(RFI_Device_Address[0]));

	RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.EN_CRC=SET;
	RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.CRCO=SET;

	RFIR_Flush_Buffer(RFIR_COM_TYPE_FLUSH_TX);
	RFIR_Flush_Buffer(RFIR_COM_TYPE_FLUSH_RX);

	/*Настройка приемника.*/
	RF_Device_Register.RFIR_RX_PW_P0_Register.RFIR_RX_PW_PX_ByteField.RX_PW_PX=sizeof(RX_Buffer
)/sizeof(RX_Buffer[0]);
	RFIR_Write_Register(RFIR_REG_TYPE_RX_PW_P0,RF_Device_Register.RFIR_RX_PW_P0_Register.byte);

	RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.MASK_RX_DR=CLEAR;
	RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.MASK_TX_DS=SET;
	RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.MASK_MAX_RT=SET;
	RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.PWR_UP=SET;
	RFIR_Write_Register(RFIR_REG_TYPE_CONFIG,RF_Device_Register.CONFIG_Register.byte);

	_delay_ms(6);

	RF_Device_Register.CONFIG_Register.RFIR_CONFIG_ByteField.PRIM_RX=SET;
	RFIR_Write_Register(RFIR_REG_TYPE_CONFIG,RF_Device_Register.CONFIG_Register.byte);

	RFIH_CE_Pin_Set();

	_delay_us(150);
}

/*****************************************************************************************
 * Прототип функции: int RFI_IRQ_Handler(void);
 * Назначение: Функция обработки прерывания от радиомодуля.
 * Аргументы: Нет.
 * Возвращаемые значения: Количество принятых байт.
 * Описание: Функция обработки прерывания от радиомодуля.
 *****************************************************************************************/
int RFI_IRQ_Handler(void)
{
	register int tmp=0;
	RFIH_CE_Pin_Clear();
	RF_Device_Register.STATUS_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_STATUS);
	if(RF_Device_Register.STATUS_Register.RFIR_STATUS_ByteField.RX_DR==SET)
	{

	RFIR_Read_Buffer(RFIR_COM_TYPE_R_RX_PAYLOAD,RX_Buffer,RF_Device_Register.RFIR_RX_PW_P0
_Register.RFIR_RX_PW_PX_ByteField.RX_PW_PX);
		tmp=sizeof(RX_Buffer)/sizeof(RX_Buffer[0]);
	}
	RFIR_Write_Register(RFIR_REG_TYPE_STATUS,RF_Device_Register.STATUS_Register.byte);
	RFIH_CE_Pin_Set();
	RFIH_IRQ_Acknowledgement();
	_delay_us(150);
	return tmp;
```

```
}

/*****************************************************************************************
 * Прототип функции: void RFI_Read_Device_Registers(void);
 * Назначение: Функция опроса регистров радиомодуля.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция опроса регистров радиомодуля.
 *****************************************************************************************/
void RFI_Read_Device_Registers(void)
{
	RF_Device_Register.CONFIG_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_CONFIG);
	RF_Device_Register.EN_AA_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_EN_AA);
	RF_Device_Register.EN_RXADDR_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_EN_RXADDR);
	RF_Device_Register.SETUP_AW_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_SETUP_AW);
	RF_Device_Register.SETUP_RETR_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_SETUP_RETR);
	RF_Device_Register.RF_CH_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RF_CH);
	RF_Device_Register.RF_SETUP_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RF_SETUP);
	RF_Device_Register.STATUS_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_STATUS);
	RF_Device_Register.OBSERVE_TX_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_OBSERVE_TX);
	RF_Device_Register.RPD_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RPD);
	RF_Device_Register.RFIR_RX_PW_P0_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RX_PW_P0);
	RF_Device_Register.RFIR_RX_PW_P1_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RX_PW_P1);
	RF_Device_Register.RFIR_RX_PW_P2_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RX_PW_P2);
	RF_Device_Register.RFIR_RX_PW_P3_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RX_PW_P3);
	RF_Device_Register.RFIR_RX_PW_P4_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RX_PW_P4);
	RF_Device_Register.RFIR_RX_PW_P5_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_RX_PW_P5);
	RF_Device_Register.FIFO_STATUS_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_FIFO_STATUS);
	RF_Device_Register.DYNPD_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_DYNPD);
	RF_Device_Register.FEATURE_Register.byte=RFIR_Read_Register(RFIR_REG_TYPE_FEATURE);
}

/*****************************************************************************************
 * Прототип функции: void RFI_Get_Recv_Data(uint8_t*,const uint8_t);
 * Назначение: Функция передачи полученных данных.
 * Аргументы: "buffer" − буффер. "buffer_length" − длина буффера.
 * Возвращаемые значения: Нет.
 * Описание: Функция передачи полученных данных.
 *****************************************************************************************/
void RFI_Get_Recv_Data(uint8_t* buffer,const uint8_t buffer_length)
{
	register uint8_t i;
	for (i=0;i<buffer_length;i++)
	{
		buffer[i]=RX_Buffer[i];
	}
}
```

Листинг заголовочного файла RFInterfaceLib.h

```
/*
 * Назначение: Заголовочный файл модуля радиоинтерфейса.
 * Имя файла: RFInterfaceLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 23.10.2014 г.
 * Список изменений:
 * 23.10.2014 г. − Исходная версия.
 */

#ifndef RFINTERFACELIB_H_
#define RFINTERFACELIB_H_

#include "GenDefineLib.h"
#include "GeneralAlgorithmsLib.h"
#include "RFInterfaceHALLib.h"
#include "RFInterfaceRELLib.h"

void RFI_Initialization(void);                                  /*Функция настройки
радиомодуля. */
int RFI_IRQ_Handler(void);                                      /*Функция обработки
прерывания от радиомодуля.*/
void RFI_Read_Device_Registers(void);                           /*Функция опроса регистров
радиомодуля.*/
void RFI_Get_Recv_Data(uint8_t*,const uint8_t);                 /*Функция передачи полученных
данных.*/

#endif /* RFINTERFACELIB_H_ */
```

Листинг файла RFInterfaceRELLib.c

```
/*
 * Назначение: Файл модуля доступа к регистрам радиоинтерфейса.
 * Имя файла: RFInterfaceRELLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 05.11.2014 г.
 * Список изменений:
 * 05.11.2014 г. − Исходная версия.
 */

#include "RFInterfaceRELLib.h"

/*Блок объявления внутренних функций.*/
uint8_t RFIR_WR_Byte(const uint8_t); /*Функция передачи/получения байта данных в/от радиомодуля.*/

/*****************************************************************************************
 * Прототип функции: uint8_t RFIR_WR_Byte(const uint8_t);
 * Назначение: Функция передачи/получения байта данных в/от радиомодуля.
 * Аргументы: "byte" − байт данных для передачи радиомодулю.
 * Возвращаемые значения: Принятый от радиомодуля байт данных.
 * Описание: Функция передачи/получения байта данных в/от радиомодуля.
 *****************************************************************************************/
uint8_t RFIR_WR_Byte(const uint8_t byte)
{
	RFIH_Write_Byte(byte);
	while(!RFIH_Check_End_Transaction());
	return RFIH_Read_Byte();
}

/*Основные функции для работы с модулем.*/

/*****************************************************************************************
 * Прототип функции: uint8_t RFIR_Read_Register(const uint8_t);
 * Назначение: Функция чтения из регистра.
 * Аргументы: "reg" − регистр.
 * Возвращаемые значения: Состояние регистра радомодуля.
 * Описание: Функция чтения данных из регистра радиомодуля.
 *****************************************************************************************/
uint8_t RFIR_Read_Register(const uint8_t reg)
{
	register uint8_t tmp;
	RFIH_CSN_Pin_Clear();
	RFIR_WR_Byte(RFIR_COM_TYPE_R_REGISTER|reg);
	tmp=RFIR_WR_Byte(0);
	RFIH_CSN_Pin_Set();
	return tmp;
}

/*****************************************************************************************
 * Прототип функции: void RFIR_Write_Register(const uint8_t,const uint8_t);
 * Назначение: Функция записи в регистр.
 * Аргументы: "reg" − регистр. "value" − записываемое значение.
 * Возвращаемые значения: Нет.
 * Описание: Функция записи данных в регистр радиомодуля.
 *****************************************************************************************/
void RFIR_Write_Register(const uint8_t reg,const uint8_t value)
{
	RFIH_CSN_Pin_Clear();
	RFIR_WR_Byte(RFIR_COM_TYPE_W_REGISTER|reg);
	RFIR_WR_Byte(value);
	RFIH_CSN_Pin_Set();
}
```

```
/*****************************************************************************************
 * Прототип функции: void RFIR_Read_Buffer(const uint8_t,uint8_t*,const uint8_t);
 * Назначение: Функция чтения массива данных из радиомодуля.
 * Аргументы: "reg" − регистр. "buffer" − буффер для прочитанной информации. \
 * "byte_num" − количество байт для чтения.
 * Возвращаемые значения: Нет.
 * Описание: Функция чтения массива данных из радиомодуля.
 *****************************************************************************************/
void RFIR_Read_Buffer(const uint8_t reg,uint8_t* buffer,const uint8_t byte_num)
{
	register uint8_t i;
	RFIH_CSN_Pin_Clear();
	RFIR_WR_Byte(RFIR_COM_TYPE_R_REGISTER|reg);
	for(i=0;i<byte_num;i++)
	{
		buffer[i]=RFIR_WR_Byte(0);
	}
	RFIH_CSN_Pin_Set();
}

/*****************************************************************************************
 * Прототип функции: void RFIR_Write_Buffer(const uint8_t,const uint8_t*,const uint8_t);
 * Назначение: Функция записи массива данных в радиомодуль.
 * Аргументы: "reg" − регистр. "buffer" − буффер для прочитанной информации. \
 * "byte_num" − количество байт для чтения.
 * Возвращаемые значения: Нет.
 * Описание: Функция записи массива данных в радиомодуль.
 *****************************************************************************************/
void RFIR_Write_Buffer(const uint8_t reg,const uint8_t* buffer,const uint8_t byte_num)
{
	register uint8_t i;
	RFIH_CSN_Pin_Clear();
	RFIR_WR_Byte(RFIR_COM_TYPE_W_REGISTER|reg);
	for(i=0;i<byte_num;i++)
	{
		RFIR_WR_Byte(buffer[i]);
	}
	RFIH_CSN_Pin_Set();
}

/*Основные комманды для работы с модулем.*/

/*****************************************************************************************
 * Прототип функции: uint8_t RFIR_Get_Status(void);
 * Назначение: Функция считывания регистра состояния радиомодуля.
 * Аргументы: Нет.
 * Возвращаемые значения: Значение регистра состояния радиомодуля.
 * Описание: Функция считывания регистра состояния радиомодуля.
 *****************************************************************************************/
uint8_t RFIR_Get_Status(void)
{
	register uint8_t tmp;
	RFIH_CSN_Pin_Clear();
	tmp=RFIR_WR_Byte(RFIR_COM_TYPE_NOP);
	RFIH_CSN_Pin_Set();
	return tmp;
}

/*****************************************************************************************
 * Прототип функции: void RFIR_Flush_Buffer(const enum RFIR_Command_TypeDef);
 * Назначение: Функция очистки буферов приема и передачи радиомодуля.
 * Аргументы: "connand" − комманда.
```

```
 * Возвращаемые значения: Нет.
 * Описание: Функция очистки буферов приема и передачи радиомодуля.
 ***********************************************************************************************/
void RFIR_Flush_Buffer(const enum RFIR_Command_TypeDef connand)
{
	RFIH_CSN_Pin_Clear();
	if(connand==RFIR_COM_TYPE_FLUSH_TX)
	{
		RFIR_WR_Byte(RFIR_COM_TYPE_FLUSH_TX);
	}
	else
	{
		RFIR_WR_Byte(RFIR_COM_TYPE_FLUSH_RX);
	}
	RFIH_CSN_Pin_Set();
}

/***********************************************************************************************
 * Прототип функции: void RFIR_Run_Write(const uint8_t);
 * Назначение: Функция запуска передачи данных.
 * Аргументы: "num" − количество передач.
 * Возвращаемые значения: Нет.
 * Описание: Функция запуска передачи данных.
 ***********************************************************************************************/
void RFIR_Run_Write(const uint8_t num)
{
	register uint8_t i;
	for(i=0;i<num;i++)
	{
		RFIH_CE_Pin_Set();
		_delay_us(20);
		RFIH_CE_Pin_Clear();
	}
}
```

Листинг заголовочного файла RFInterfaceRELLib.h

```
/*
 * Назначение: Заголовочный файл модуля доступа к регистрам радиоинтерфейса.
 * Имя файла: RFInterfaceRELLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 05.11.2014 г.
 * Список изменений:
 * 05.11.2014 г. − Исходная версия.
 */

#ifndef RFINTERFACERELLIB_H_
#define RFINTERFACERELLIB_H_

#include "RFInterfaceHALLib.h"

/*Команды для взаимодействия с радомодулем.*/
enum RFIR_Command_TypeDef
{
 RFIR_COM_TYPE_R_REGISTER=0x00, /*Прочитать из регистра.*/
 RFIR_COM_TYPE_W_REGISTER=0x20, /*Записать в регистр.*/
 RFIR_COM_TYPE_R_RX_PAYLOAD=0x61, /*Прочитать полученную полезную нагрузку.*/
 RFIR_COM_TYPE_W_TX_PAYLOAD=0xA0, /*Записать передаваемую полезную нагрузку.*/
 RFIR_COM_TYPE_FLUSH_TX=0xE1, /*Очистить буфер передачи.*/
 RFIR_COM_TYPE_FLUSH_RX=0xE2, /*Очистить буфер приема.*/
 RFIR_COM_TYPE_REUSE_TX_PL=0xE3, /*Повторить передачу полезной нагрузки из буфера передачи.*/
 RFIR_COM_TYPE_R_RX_PL_WID=0x60, /*Считать принятую полезную нагрузку с вершины стека.*/
 RFIR_COM_TYPE_W_ACK_PAYLOAD=0xA8, /*Записать полезную нагрузку передаваемую в фазе
подтверждения полученного пакета.*/
 RFIR_COM_TYPE_W_TX_PAYLOAD_NOACK=0xB0, /*Записать полезную нагузку с отменой передачи полезной
нагрузки в фазе подтверждения полученного пакета.*/
 RFIR_COM_TYPE_NOP=0xFF, /*Прочитать сегистр состояния радиомодуля.*/
};

/**/
enum RFIR_Register_TypeDef
{
 RFIR_REG_TYPE_CONFIG=0x00, /*Адрес регистра "CONFIG".*/
 RFIR_REG_TYPE_EN_AA=0x01, /*Адрес регистра "EN_AA".*/
 RFIR_REG_TYPE_EN_RXADDR=0x02, /*Адрес регистра "RXADDR".*/
 RFIR_REG_TYPE_SETUP_AW=0x03, /*Адрес регистра "SETUP_AW".*/
 RFIR_REG_TYPE_SETUP_RETR=0x04, /*Адрес регистра "SETUP_RETR".*/
 RFIR_REG_TYPE_RF_CH=0x05, /*Адрес регистра "RF_CH".*/
 RFIR_REG_TYPE_RF_SETUP=0x06, /*Адрес регистра "RF_SETUP".*/
 RFIR_REG_TYPE_STATUS=0x07, /*Адрес регистра "STATUS".*/
 RFIR_REG_TYPE_OBSERVE_TX=0x08, /*Адрес регистра "OBSERVE_TX".*/
 RFIR_REG_TYPE_RPD=0x09, /*Адрес регистра "RPD".*/
 RFIR_REG_TYPE_RX_ADDR_P0=0x0A, /*Адрес регистра "RX_ADDR_P0".*/
 RFIR_REG_TYPE_RX_ADDR_P1=0x0B, /*Адрес регистра "RX_ADDR_P1".*/
 RFIR_REG_TYPE_RX_ADDR_P2=0x0C, /*Адрес регистра "RX_ADDR_P2".*/
 RFIR_REG_TYPE_RX_ADDR_P3=0x0D, /*Адрес регистра "RX_ADDR_P3".*/
 RFIR_REG_TYPE_RX_ADDR_P4=0x0E, /*Адрес регистра "RX_ADDR_P4".*/
 RFIR_REG_TYPE_RX_ADDR_P5=0x0F, /*Адрес регистра "RX_ADDR_P5".*/
 RFIR_REG_TYPE_TX_ADDR=0x10, /*Адрес регистра "TX_ADDR".*/
 RFIR_REG_TYPE_RX_PW_P0=0x11, /*Адрес регистра "RX_PW_P0".*/
 RFIR_REG_TYPE_RX_PW_P1=0x12, /*Адрес регистра "RX_PW_P1".*/
 RFIR_REG_TYPE_RX_PW_P2=0x13, /*Адрес регистра "RX_PW_P2".*/
 RFIR_REG_TYPE_RX_PW_P3=0x14, /*Адрес регистра "RX_PW_P3".*/
 RFIR_REG_TYPE_RX_PW_P4=0x15, /*Адрес регистра "RX_PW_P4".*/
 RFIR_REG_TYPE_RX_PW_P5=0x16, /*Адрес регистра "RX_PW_P5".*/
 RFIR_REG_TYPE_FIFO_STATUS=0x17, /*Адрес регистра "FIFO_STATUS".*/
 RFIR_REG_TYPE_DYNPD=0x1C, /*Адрес регистра "DYNPD".*/
 RFIR_REG_TYPE_FEATURE=0x1D, /*Адрес регистра "FEATURE".*/
```

```
};

/*Структура регистра "CONFIG".*/
struct RFIR_CONFIG_ByteField_TypeDef
{
 uint8_t PRIM_RX:1; /*Бит режима работы радиомодуля приемник/передатчик.*/
 uint8_t PWR_UP:1; /*Включение питания радиомодуля.*/
 uint8_t CRCO:1; /*Контрольная сумма длиной 1 или 2 байта.*/
 uint8_t EN_CRC:1; /*Включение контрольной суммы.*/
 uint8_t MASK_MAX_RT:1; /*Включение/выключение прерывания MAX_RT.*/
 uint8_t MASK_TX_DS:1; /*Включение/выключение прерывания TX_DS.*/
 uint8_t MASK_RX_DR:1; /*Включение/выключение прерывания RX_DR.*/
 uint8_t RESERVED:1; /*Зарезервировано.*/
};

/*Структура регистра "EN_AA".*/
struct RFIR_EN_AA_ByteField_TypeDef
{
 uint8_t ENAA_P0:1; /*Включение/выключение автоматического подтверждения принятого пакета логического
канала 0.*/
 uint8_t ENAA_P1:1; /*Включение/выключение автоматического подтверждения принятого пакета логического
канала 1.*/
 uint8_t ENAA_P2:1; /*Включение/выключение автоматического подтверждения принятого пакета логического
канала 2.*/
 uint8_t ENAA_P3:1; /*Включение/выключение автоматического подтверждения принятого пакета логического
канала 3.*/
 uint8_t ENAA_P4:1; /*Включение/выключение автоматического подтверждения принятого пакета логического
канала 4.*/
 uint8_t ENAA_P5:1; /*Включение/выключение автоматического подтверждения принятого пакета логического
канала 5.*/
 uint8_t RESERVED:2; /*Зарезервировано.*/
};

/*Структура регистра "EN_RXADDR".*/
struct RFIR_EN_RXADDR_ByteField_TypeDef
{
 uint8_t ERX_P0:1; /*Включение/выключение логического канала 0.*/
 uint8_t ERX_P1:1; /*Включение/выключение логического канала 1.*/
 uint8_t ERX_P2:1; /*Включение/выключение логического канала 2.*/
 uint8_t ERX_P3:1; /*Включение/выключение логического канала 3.*/
 uint8_t ERX_P4:1; /*Включение/выключение логического канала 4.*/
 uint8_t ERX_P5:1; /*Включение/выключение логического канала 5.*/
 uint8_t RESERVED:2; /*Зарезервировано.*/
};

/*Структура регистра "SETUP_AW".*/
struct RFIR_SETUP_AW_ByteField_TypeDef
{
 uint8_t AW:2; /*Ширина поля адреса.*/
 uint8_t RESERVED:6; /*Зарезервировано.*/
};

/*Структура регистра "SETUP_RETR".*/
struct RFIR_SETUP_RETR_ByteField_TypeDef
{
 uint8_t ARC:4; /*Счетчик автоматических повторных передачь.*/
 uint8_t ARD:4; /*Время автоматической задержки передачи.*/
};

/*Структура регистра "RF_CH".*/
struct RFIR_RF_CH_ByteField_TypeDef
{
 uint8_t RF_CH:7; /*Радиоканал.*/
```

```
 uint8_t RESERVED:1; /*Зарезервировано.*/
};

/*Структура регистра "RF_SETUP".*/
struct RFIR_RF_SETUP_ByteField_TypeDef
{
 uint8_t Obsolete:1; /*Не используется.*/
 uint8_t RF_PWR:2; /*Мощность выходного сигнала.*/
 uint8_t RF_DR_HIGH:1; /*Скорость 1 или 2 МБит/с.*/
 uint8_t PLL_LOCK:1; /*Включение блокировки ФАПЧ. Только для тестового режима.*/
 uint8_t RF_DR_LOW:1; /*Скорость 250 кБит/с.*/
 uint8_t RESERVED:1; /*Зарезервировано.*/
 uint8_t CONT_WAVE:1; /*Включение несущей. Только для тестового режима.*/
};

/*Структура регистра "STATUS".*/
struct RFIR_STATUS_ByteField_TypeDef
{
 uint8_t TX_FULL:1; /*Флаг заполнения буфера передатчика.*/
 uint8_t RX_P_NO:3; /*Номер логического канала в котором приняты данные.*/
 uint8_t MAX_RT:1; /*Флаг прерывания достижения максимального количества повторных передач.*/
 uint8_t TX_DS:1; /*Флаг прерывания завершения передачи данных.*/
 uint8_t RX_DR:1; /*Флаг прерывания завершения приема данных.*/
 uint8_t RESERVED:1; /*Зарезервировано.*/
};

/*Структура регистра "OBSERVE_TX".*/
struct RFIR_OBSERVE_TX_ByteField_TypeDef
{
 uint8_t ARC_CNT:4; /*Счетчик повторно переданных пакетов.*/
 uint8_t PLOS_CNT:4; /*Счетчик патеряных пакетов.*/
};

/*Структура регистра "RPD".*/
struct RFIR_RPD_ByteField_TypeDef
{
 uint8_t RPD:1; /*Бит приемника детектора мощности.*/
 uint8_t RESERVED:7; /*Зарезервировано.*/
};

/*Структура регистра "RX_PW_PX".*/
struct RFIR_RX_PW_PX_ByteField_TypeDef
{
 uint8_t RX_PW_PX:6; /*Количество принятых байт в логическом канале X.*/
 uint8_t RESERVED:2; /*Зарезервировано.*/
};

/*Структура регистра "FIFO_STATUS".*/
struct RFIR_FIFO_STATUS_ByteField_TypeDef
{
 uint8_t RX_EMPTY:1; /*Флаг опустошения буфера приемника.*/
 uint8_t RX_FULL:1; /*Флаг заполнения буфера приемника.*/
 uint8_t RESERVED1:2; /*Зарезервировано.*/
 uint8_t TX_EMPTY:1; /*Флаг опустошения буфера передатчика.*/
 uint8_t TX_FULL:1; /*Флаг заполнения буфера передатчика.*/
 uint8_t TX_REUSE:1; /*Передача последнеиспользуемой полезной нагрузки.*/
 uint8_t RESERVED2:1; /*Зарезервировано.*/
};

/*Структура регистра "DYNPD".*/
struct RFIR_DYNPD_ByteField_TypeDef
{
 uint8_t DPL_P0:1; /*Включение динамической полезной нагрузки для логического канала 0.*/
```

```
 uint8_t DPL_P1:1; /*Включение динамической полезной нагрузки для логического канала 1.*/
 uint8_t DPL_P2:1; /*Включение динамической полезной нагрузки для логического канала 2.*/
 uint8_t DPL_P3:1; /*Включение динамической полезной нагрузки для логического канала 3.*/
 uint8_t DPL_P4:1; /*Включение динамической полезной нагрузки для логического канала 4.*/
 uint8_t DPL_P5:1; /*Включение динамической полезной нагрузки для логического канала 5.*/
 uint8_t RESERVED:2; /*Зарезервировано.*/
};

/*Структура регистра "FEATURE".*/
struct RFIR_FEATURE_ByteField_TypeDef
{
 uint8_t EN_DYN_ACK:1; /*Включение команды "RFIR_COM_TYPE_W_TX_PAYLOAD_NOACK".*/
 uint8_t EN_ACK_PAY:1; /*Включение подтверждения пакета с передачей полезной нагрузкой.*/
 uint8_t EN_DPL:1; /*Включение динамической длины полезной нагрузки.*/
 uint8_t RESERVED:5; /*Зарезервировано.*/
};

/*Описание регистра "CONFIG".*/
union RFIR_CONFIG_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "CONFIG".*/
 struct RFIR_CONFIG_ByteField_TypeDef RFIR_CONFIG_ByteField; /*Структура регистра "CONFIG".*/
};

/*Описание регистра "EN_AA".*/
union RFIR_EN_AA_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "EN_AA".*/
 struct RFIR_EN_AA_ByteField_TypeDef RFIR_EN_AA_ByteField; /*Структура регистра "EN_AA".*/
};

/*Описание регистра "EN_RXADDR".*/
union RFIR_EN_RXADDR_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "EN_RXADDR".*/
 struct RFIR_EN_RXADDR_ByteField_TypeDef RFIR_EN_RXADDR_ByteField; /*Структура регистра
"EN_RXADDR".*/
};

/*Описание регистра "SETUP_AW".*/
union RFIR_SETUP_AW_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "SETUP_AW".*/
 struct RFIR_SETUP_AW_ByteField_TypeDef RFIR_SETUP_AW_ByteField; /*Структура регистра "SETUP_AW".*/
};

/*Описание регистра "SETUP_RETR".*/
union RFIR_SETUP_RETR_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "SETUP_RETR".*/
 struct RFIR_SETUP_RETR_ByteField_TypeDef RFIR_SETUP_RETR_ByteField; /*Структура регистра
"SETUP_RETR".*/
};

/*Описание регистра "RF_CH".*/
union RFIR_RF_CH_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "RF_CH".*/
 struct RFIR_RF_CH_ByteField_TypeDef RFIR_RF_CH_ByteField; /*Структура регистра "RF_CH".*/
};

/*Описание регистра "RF_SETUP".*/
union RFIR_RF_SETUP_Register_TypeDef
```

```
{
 uint8_t byte; /*Байт регистра "RF_SETUP".*/
 struct RFIR_RF_SETUP_ByteField_TypeDef RFIR_RF_SETUP_ByteField; /*Структура регистра "RF_SETUP".*/
};

/*Описание регистра "STATUS".*/
union RFIR_STATUS_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "STATUS".*/
 struct RFIR_STATUS_ByteField_TypeDef RFIR_STATUS_ByteField; /*Структура регистра "STATUS".*/
};

/*Описание регистра "OBSERVE_TX".*/
union RFIR_OBSERVE_TX_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "OBSERVE_TX".*/
 struct RFIR_OBSERVE_TX_ByteField_TypeDef RFIR_OBSERVE_TX_ByteField; /*Структура регистра
"OBSERVE_TX".*/
};

/*Описание регистра "RPD".*/
union RFIR_RPD_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "RPD".*/
 struct RFIR_RPD_ByteField_TypeDef RFIR_RPD_ByteField; /*Структура регистра "RPD".*/
};

/*Описание регистра "RX_PW_PX".*/
union RFIR_RX_PW_PX_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "RX_PW_PX".*/
 struct RFIR_RX_PW_PX_ByteField_TypeDef RFIR_RX_PW_PX_ByteField; /*Структура регистра "RX_PW_PX".*/
};

/*Описание регистра "FIFO_STATUS".*/
union RFIR_FIFO_STATUS_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "FIFO_STATUS".*/
 struct RFIR_FIFO_STATUS_ByteField_TypeDef RFIR_FIFO_STATUS_ByteField; /*Структура регистра
"FIFO_STATUS".*/
};

/*Описание регистра "DYNPD".*/
union RFIR_DYNPD_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "DYNPD".*/
 struct RFIR_DYNPD_ByteField_TypeDef RFIR_DYNPD_ByteField; /*Структура регистра "DYNPD".*/
};

/*Описание регистра "FEATURE".*/
union RFIR_FEATURE_Register_TypeDef
{
 uint8_t byte; /*Байт регистра "FEATURE".*/
 struct RFIR_FEATURE_ByteField_TypeDef RFIR_FEATURE_ByteField; /*Структура регистра "FEATURE".*/
};

/*Регистры радомодуля.*/
struct RFIR_Registers_TypeDef
{
 union RFIR_CONFIG_Register_TypeDef CONFIG_Register; /*Регистр "CONFIG".*/
 union RFIR_EN_AA_Register_TypeDef EN_AA_Register; /*Регистр "EN_AA".*/
 union RFIR_EN_RXADDR_Register_TypeDef EN_RXADDR_Register; /*Регистр "EN_RXADDR".*/
 union RFIR_SETUP_AW_Register_TypeDef SETUP_AW_Register; /*Регистр "SETUP_AW".*/
```

```
 union RFIR_SETUP_RETR_Register_TypeDef SETUP_RETR_Register; /*Регистр "SETUP_RETR".*/
 union RFIR_RF_CH_Register_TypeDef RF_CH_Register; /*Регистр "RF_CH".*/
 union RFIR_RF_SETUP_Register_TypeDef RF_SETUP_Register; /*Регистр "RF_SETUP".*/
 union RFIR_STATUS_Register_TypeDef STATUS_Register; /*Регистр "STATUS".*/
 union RFIR_OBSERVE_TX_Register_TypeDef OBSERVE_TX_Register; /*Регистр "OBSERVE_TX".*/
 union RFIR_RPD_Register_TypeDef RPD_Register; /*Регистр "RPD".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P0_Register; /*Регистр "RFIR_RX_PW_P0".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P1_Register; /*Регистр "RFIR_RX_PW_P1".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P2_Register; /*Регистр "RFIR_RX_PW_P2".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P3_Register; /*Регистр "RFIR_RX_PW_P3".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P4_Register; /*Регистр "RFIR_RX_PW_P4".*/
 union RFIR_RX_PW_PX_Register_TypeDef RFIR_RX_PW_P5_Register; /*Регистр "RFIR_RX_PW_P5".*/
 union RFIR_FIFO_STATUS_Register_TypeDef FIFO_STATUS_Register; /*Регистр "FIFO_STATUS".*/
 union RFIR_DYNPD_Register_TypeDef DYNPD_Register; /*Регистр "DYNPD".*/
 union RFIR_FEATURE_Register_TypeDef FEATURE_Register; /*Регистр "FEATURE".*/
};

/*Основные функции для работы с модулем.*/
uint8_t RFIR_Read_Register(const uint8_t); /*Функция чтения из регистра.*/
void RFIR_Write_Register(const uint8_t,const uint8_t); /*Функция записи в регистр.*/
void RFIR_Read_Buffer(const uint8_t,uint8_t*,const uint8_t); /*Функция чтения массива данных из радиомодуля.*/
void RFIR_Write_Buffer(const uint8_t,const uint8_t*,const uint8_t);   /*Функция записи массива данных в
радиомодуль. */

/*Основные комманды для работы с модулем.*/
uint8_t RFIR_Get_Status(void);
        /*Функция считывания регистра состояния радиомодуля.*/
void RFIR_Flush_Buffer(const enum RFIR_Command_TypeDef);                    /*Функция очистки буферов
приема и передачи радиомодуля.*/
void RFIR_Run_Write(const uint8_t);
        /*Функция запуска передачи данных.*/

#endif /* RFINTERFACERELLIB_H_ */
```

Листинг файла RTOSHALLib.c

```
/*
 * Назначение: Файл модуля абстракции ОСРВ от аппаратуры.
 * Имя файла: RTOSHALLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 17.10.2014 г.
 * Список изменений:
 * 17.01.2014 г. − Исходная версия.
*/

#include "RTOSHALLib.h"

/*****************************************************************************
************
 * Прототип функции: void RTOSHALTaskTimerInitialization(const uint16_t,const uint16_t);
 * Назначение: Функция настройки системного таймера задач для МК.
 * Аргументы: "psc" − значение предделителя таймера. "arr" − значение счетчика таймера при
 * достижении которого генерируется прерывание.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки системного таймера задач для МК.

*****************************************************************************
***********/
void RTOSHALTaskTimerInitialization(const uint16_t psc,const uint16_t arr)
{
	if(psc>0)
	{
		BREAK;	/*Включить предделитель.*/
	}
	OCR1A=arr;
	SET_BIT(TIMSK,OCIE1A);
	TCCR1B=(1<<WGM12)|(1<<CS10);
}
```

Листинг заголовочного файла RTOSHALLib.h

```
/*
 * Назначение: Заголовочный файл модуля абстракции ОСРВ от аппаратуры.
 * Имя файла: RTOSHALLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 17.10.2014 г.
 * Список изменений:
 * 17.01.2014 г. − Исходная версия.
*/

#ifndef RTOSHALLIB_H_
#define RTOSHALLIB_H_

#include "GenDefineLib.h"

#define RTOS_MAX_TIME_COUNTER_VALUE CONFIG_MAX_TIME_COUNTER_VALUE
        /*Максимальное число помещающееся в регистре таймера.*/
#define RTOS_TIME_QUANT CONFIG_TIME_QUANT
                /*Квант времени планировщика в с.*/
#define RTOS_MCU_FREQUENCY CONFIG_MCU_FREQUENCY
                /*Тактовая частота таймера в Гц.*/

void RTOSHALTaskTimerInitialization(const uint16_t,const uint16_t);          /*Функция
настройки системного таймера задач для МК.*/

#endif /* RTOSHALLIB_H_ */
```

Листинг файла RTOSShedulerLib.c

```
/*
 * Назначение: Файл модуля планировщика задач ОСРВ.
 * Имя файла: RTOSShedulerLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 17.01.2014 г.
 * Список изменений:
 * 17.01.2014 г. − Исходная версия.
 * 24.03.2014 г. − Исполняемые задачи описываются массивом структур.
 * 06.06.2014 г. − Структуры планировщика и исполняемые задачи внесены в модуль планировщика.
 * 17.10.2014 г. − Изменена структура планировщика.
*/

#include "RTOSShedulerLib.h"

struct RTOSTask_TypeDef RTOSSystemTask[]=
{
	{
		RTOS_TIME_TASK_0,
		RTOS_TIME_TASK_0,
		RTOS_TASK_STATUS_REDY,
	},
};

int RFRXDataNum;	/*Количество принятых байт данных.*/

/*Блок определений внутренних функций.*/
void RTOSInitialization(void); /*Функция настройки временного кванта планировщика.*/
void RTOSTaskFunction0(void); /*Функция выполнения задачи 0.*/

/*****************************************************************************************
 * Прототип функции: static void RTOSInitialization(void);
 * Назначение: Функция настройки временного кванта планировщика.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция задает минимальное квантование времени для планировщика задач.
 *****************************************************************************************/
void RTOSInitialization(void)
{
 unsigned int tmp=RTOS_TIME_QUANT*RTOS_MCU_FREQUENCY;
 unsigned int i=0;
 if(tmp>RTOS_MAX_TIME_COUNTER_VALUE)
 {
 unsigned int j=tmp;
 for(i=2;tmp>RTOS_MAX_TIME_COUNTER_VALUE;i++)
 {
 if((tmp/=i)>RTOS_MAX_TIME_COUNTER_VALUE)
 {
 tmp=j;
 }
 }
 }
 RTOSHALTaskTimerInitialization((uint16_t)i,(uint16_t)tmp);
}

/*****************************************************************************************
 * Прототип функции: void RTOSTaskCounter(void);
 * Назначение: Функция инкремента счетчика задач.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция инкремента счетчика временных квантов оставшихся до исполнения задачи.
 *****************************************************************************************/
```

```
void RTOSTaskCounter(void)
{
 struct RTOSTask_TypeDef* first_element=&RTOSSystemTask[0];
 struct RTOSTask_TypeDef*
last_element=&RTOSSystemTask[(sizeof(RTOSSystemTask)/sizeof(RTOSSystemTask[0]))];
 while(first_element<last_element)
 {
 if(first_element->Status==RTOS_TASK_STATUS_REDY)
 {
 if((--first_element->TaskTimeCounter)==0)
 {
 first_element->TaskTimeCounter=first_element->TaskTime;
 first_element->Status=RTOS_TASK_STATUS_EXECUTE;
 }
 }
 first_element++;
 }
}

/*****************************************************************************************
 * Прототип функции: void RTOSTaskHandler(void);
 * Назначение: Функция запуска обработчиков задач.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция запускает и останавливает обработчики задач. Должна распологаться в
 * основном цикле main.
 *****************************************************************************************/
void RTOSTaskHandler(void)
{
 struct RTOSTask_TypeDef* first_element=&RTOSSystemTask[0];
 struct RTOSTask_TypeDef*
last_element=&RTOSSystemTask[(sizeof(RTOSSystemTask)/sizeof(RTOSSystemTask[0]))];
 register unsigned int i=0;
 while(first_element<last_element)
 {
 if(first_element->Status==RTOS_TASK_STATUS_EXECUTE)
 {
 switch(i)
 {
                    case 0:
 {
                              RTOSTaskFunction0();
                              break;
 }
                    default:
                    {
                              break;
                    }
 }
 first_element->Status=RTOS_TASK_STATUS_REDY;
 }
 i++;
 first_element++;
 }
 /*Блок постоянного выполнения.*/
 {
 if(RFRXDataNum>0)
 {
                    uint8_t buffer[8];
          RFI_Get_Recv_Data(&buffer[0],RFRXDataNum);
          if ((buffer[0]==START_BYTE)&&(buffer[7]==STOP_BYTE))
          {
                              if ((buffer[1]==PREAMBULE_BYTE_1)||(buffer[1]==PREAMBULE_BYTE_2))
```

```
                        {
                                if (buffer[6]==GetXor(&buffer[2],&buffer[5]))
                                {
                                        UI_UART_Send_Data(ByteToUint16_t(&buffer[2]), buffer[4], buffer[5]);
                                }
                        }
        }
                RFRXDataNum=0;
 }
 }
}

/*****************************************************************************************
 * Прототип функции: void RTOSHardwareInitialization(void);
 * Назначение: Функция настройки аппаратуры.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настройки аппаратуры.
 *****************************************************************************************/
void RTOSHardwareInitialization(void)
{
        RTOSInitialization();
 RFI_Initialization();
        UI_UART_Initialization();
        HIDIInitialization();
        sei();
        HIDILEDSetState(GENERAL_STATE_ENABLE);
}

/*****************************************************************************************
 * Прототип функции: void RTOSIRQHandler(enum RTOSInterrupt_TypeDef);
 * Назначение: Функция вызова обработчиков прерываний от аппаратуры.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция вызова обработчиков прерываний от аппаратуры.
 *****************************************************************************************/
void RTOSIRQHandler(enum RTOSInterrupt_TypeDef RTOSInterrupt)
{
        switch(RTOSInterrupt)
        {
                case RTOS_IRQ_HW_RF_MODULE:
                {
                        RFRXDataNum=RFI_IRQ_Handler();
                        break;
                }
                case RTOS_IRQ_HW_UART:
                {
                        UI_UART_IRQ_Handler();
                        break;
                }
                default:
                {
                        break;
                }
        }
}

/*****************************************************************************************
* Прототип: static void RTOSTaskFunction0(void);
* Назначение: Функция выполнения задачи 0.
* Аргументы: Нет.
* Возвращаемое значение: Нет.
* Описание: Функция выполнения задачи 0.
```

```
*****************************************************************************************/
void RTOSTaskFunction0(void)
{
	if (HIDILEDGetState()==GENERAL_STATE_ENABLE)
	{
		HIDILEDSetState(GENERAL_STATE_DISABLE);
	}
}
```

Листинг заголовочного файла RTOSShedulerLib.h

```
/*
 * Назначение: Заголовочный файл модуля планировщика задач ОСРВ.
 * Имя файла: RTOSShedulerLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 17.01.2014 г.
 * Список изменений:
 * 17.01.2014 г. − Исходная версия.
 * 24.03.2014 г. − Исполняемые задачи описываются массивом структур.
 * 06.06.2014 г. − Структуры планировщика и исполняемые задачи внесены в модуль планировщика.
 * 17.10.2014 г. − Изменена структура планировщика.
 */

#ifndef RTOSSHEDULERLIB_H_
#define RTOSSHEDULERLIB_H_

#include "RTOSHALLib.h"
#include "RFInterfaceLib.h"
#include "UARTInterfaceLib.h"
#include "HIDInterfaceLib.h"

#define RTOS_TIME_TASK_0     (CONFIG_LED_BLINK/CONFIG_TIME_QUANT)            /*Период исполнения
задачи 0, исчисляется в квантах времени планировщика.*/

/*Режимы состояния выполнения задачи.*/
enum RTOSTaskStatus_TypeDef
{
 RTOS_TASK_STATUS_REDY, /*Выполнение задачи запрещено.*/
 RTOS_TASK_STATUS_EXECUTE, /*Выполнение задачи разрешено.*/
};

/*Прерывания с которым работает ОСРВ.*/
enum RTOSInterrupt_TypeDef
{
        RTOS_IRQ_HW_RF_MODULE,  /*Прерывание от радиомодуля.*/
        RTOS_IRQ_HW_UART,       /*Прерывание от UART.*/
        RTOS_IRQ_HW_NULL,       /*Отсутствие прерываний от аппаратуры.*/
        RTOS_IRQ_SW_NULL,       /*Отсутствие прерываний от программных модулей.*/
};

/*Состояние данных в буфере.*/
enum RTOSDataStatus_TypeDef
{
 RTOS_DATA_STATUS_OLD, /*Данные не актуальны.*/
 RTOS_DATA_STATUS_NEW, /*Данные актуальны.*/
 RTOS_DATA_STATUS_ERROR, /*В данных ошибка.*/
 RTOS_DATA_STATUS_NULL, /*Данные не приняты.*/
};

/*Структура описания задачи.*/
struct RTOSTask_TypeDef
{
 const unsigned int TaskTime; /*Период исполнениязадачи в минимальных квантах времени ОСРВ.*/
 unsigned int TaskTimeCounter; /*Счетчик временных квантов оставшихся до исполнения задачи.*/
 enum RTOSTaskStatus_TypeDef Status; /*Статус состояния выполнения задачи.*/
};

void RTOSHardwareInitialization(void); /*Функция настройки аппаратуры.*/
void RTOSIRQHandler(enum RTOSInterrupt_TypeDef); /*Функция вызова обработчиков прерываний от
аппаратуры.*/
void RTOSTaskCounter(void); /*Функция инкремента счетчика задач.*/
void RTOSTaskHandler(void); /*Функция запуска обработчиков задач.*/
```

```
#endif /* RTOSSHEDULERLIB_H_ */
```

Листинг файла UARTInterfaceHALLib.c

```
/*
 * Назначение: Файл модуля абстракции UART от аппаратуры.
 * Имя файла: UARTInterfaceHALLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 02.12.2014 г.
 * Список изменений:
 * 02.12.2014 г. − Исходная версия.
 */

#include "UARTInterfaceHALLib.h"

/*****************************************************************************
************
 * Прототип функции: void UIH_UART_Init(void);
 * Назначение: Функция настраивает асинхронный приемопередатчик.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настраивает асинхронный приемопередатчик на скорость 9600 бод, \
 * 1 стоп−бит, без контроля четности и без апаратного управления потоком. При частоте ядра \
 * 8 МГц.

*****************************************************************************
***********/
void UIH_UART_Init(void)
{
	WRITE_REG(UBRRL,LOW(UIH_BAUD_VALUE));
	WRITE_REG(UBRRH,HIGH(UIH_BAUD_VALUE));
	SET_BIT(UCSRB,TXEN);
	WRITE_REG(UCSRC,((1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1))); /*При отладке не
отображается.*/
	SET_BIT(DDRD,PD3);
	SET_BIT(PORTD,PD3);
}
```

Листинг заголовочного файла UARTInterfaceHALLib.h

```
/*
 * Назначение: Заголовочный файл модуля абстракции UART от аппаратуры.
 * Имя файла: UARTInterfaceHALLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 02.12.2014 г.
 * Список изменений:
 * 02.12.2014 г. − Исходная версия.
 */

#ifndef UARTINTERFACEHALLIB_H_
#define UARTINTERFACEHALLIB_H_

#include "GenDefineLib.h"

#define UIH_XTAL CONFIG_MCU_FREQUENCY                                   /*Частота
ядра в Гц.*/
#define UIH_BAUDRATE CONFIG_UART_BAUDRATE                               /*Скорость
передачи в бод.*/
#define UIH_BAUD_VALUE (UIH_XTAL/(16*UIH_BAUDRATE)−1) /*Числовое значение
скорости передачи.*/

#define UIH_UART_Transfer_Enable() SET_BIT(UCSRB,UDRIE)                 /*Разрешить
передачу байта по УАПП.*/
#define UIH_UART_Transfer_Disable() CLEAR_BIT(UCSRB,UDRIE) /*Запретить передачу байта
по УАПП.*/
#define UIH_UART_Write_Data(byte)       WRITE_REG(UDR,byte)             /*Записать байт для
передачи.*/

void UIH_UART_Init(void); /*Функция настраивает асинхронный приемопередатчик.*/

#endif /* UARTINTERFACEHALLIB_H_ */
```

Листинг файла UARTInterfaceLib.c

```
/*
 * Назначение: Файл модуля UART.
 * Имя файла: UARTInterfaceLib.c
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 02.12.2014 г.
 * Список изменений:
 * 02.12.2014 г. − Исходная версия.
 */

#include "UARTInterfaceLib.h"

uint8_t UI_UART_TX_Buffer[UI_UART_BUFFER_LENGTH];
	/*Буффер передачи УАПП.*/
uint8_t UI_UART_Counter=0;
	/*Счетчик переданных байт.*/
enum GeneralState_TypeDef UI_UART_Write_Buf_State=GENERAL_STATE_TRUE;		/*Ключ разрешения
записи новых данных.*/

/*****************************************************************************************
 * Прототип функции: void UI_UART_Initialization(void);
 * Назначение: Функция настраивает УАПП.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция настраивает УАПП.
 *****************************************************************************************/
void UI_UART_Initialization(void)
{
	UIH_UART_Init();
}

/*****************************************************************************************
 * Прототип функции: void UI_UART_IRQ_Handler(void);
 * Назначение: Функция обработки прерывания от УАПП.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция обработки прерывания от УАПП.
 *****************************************************************************************/
void UI_UART_IRQ_Handler(void)
{
	register uint8_t tmp=UI_UART_Counter;
	if (tmp<(sizeof(UI_UART_TX_Buffer)/sizeof(UI_UART_TX_Buffer[0])))
	{
		UIH_UART_Write_Data(UI_UART_TX_Buffer[tmp++]);
	}
	else
	{
		tmp=0;
		UI_UART_Write_Buf_State=GENERAL_STATE_TRUE;
		UIH_UART_Transfer_Disable();
	}
	UI_UART_Counter=tmp;
}

/*****************************************************************************************
 * Прототип функции: int UI_UART_Send_Data(const uint16_t,const uint8_t, const uint8_t);
 * Назначение: Функция передачи пакета информации на хост.
 * Аргументы: "data" − данные с датчика. "resist" − данные о сопротивлении с датчика.
 * Возвращаемые значения: "0" − успешная передача. "−1" − ошибка.
 * Описание: Функция передачи пакета информации на хост.
 *****************************************************************************************/
int UI_UART_Send_Data(const uint16_t data, const uint8_t resist, const uint8_t counter)
```

```
{
	static enum GeneralState_TypeDef preambule_state = GENERAL_STATE_FALSE;
	register int tmp=0;
	register uint8_t preambule;
	if (UI_UART_Write_Buf_State==GENERAL_STATE_TRUE)
	{
		UI_UART_Write_Buf_State=GENERAL_STATE_FALSE;
		UI_UART_TX_Buffer[0]=START_BYTE;
		if(preambule_state==GENERAL_STATE_FALSE)
		{
			preambule_state=GENERAL_STATE_TRUE;
			preambule=PREAMBULE_BYTE_1;
		}
		else
		{
			preambule_state=GENERAL_STATE_FALSE;
			preambule=PREAMBULE_BYTE_2;
		}
		UI_UART_TX_Buffer[1]=preambule;
		Uint16_tToByte(&UI_UART_TX_Buffer[2],data);
		UI_UART_TX_Buffer[4]=resist;
		UI_UART_TX_Buffer[5]=counter;
		UI_UART_TX_Buffer[6]=GetXor(&UI_UART_TX_Buffer[2],&UI_UART_TX_Buffer[5]);
		UI_UART_TX_Buffer[7]=STOP_BYTE;
		tmp=UI_UART_BUFFER_LENGTH;
		UIH_UART_Transfer_Enable();
	}
	else
	{
		tmp=−1;
	}
	return tmp;
}
```

Листинг заголовочного файла UARTInterfaceLib.h

```
/*
 * Назначение: Заголовочный файл модуля UART.
 * Имя файла: UARTInterfaceLib.h
 * Автор: Сидоренко Михаил Сергеевич, mikes.sidorenko@gmail.com
 * Создан: 02.12.2014 г.
 * Список изменений:
 * 02.12.2014 г. − Исходная версия.
 */

#ifndef UARTINTERFACELIB_H_
#define UARTINTERFACELIB_H_

#include "UARTInterfaceHALLib.h"
#include "GeneralAlgorithmsLib.h"

#define UI_UART_BUFFER_LENGTH CONFIG_UART_BUFFER_LENGTH /*Максимальная
длина буфера данных.*/

void UI_UART_Initialization(void);                                  /*Функция настраивает
УАПП.*/
void UI_UART_IRQ_Handler(void);                                     /*Функция
обработки прерывания от УАПП.*/
int UI_UART_Send_Data(const uint16_t, const uint8_t, const uint8_t);   /*Функция передачи пакета
информации на хост.*/

#endif /* UARTINTERFACELIB_H_ */
```

ЛИСТИНГ СПО

Листинг файла Program.cs

```
/*
 * Назначение: Модуль основного файла программы.
 * Имя файла: Program.cs
 * Автор: Сидоренко Михаил Сергеевич
 * Создан: 14.04.2015
 * Список изменений: 14.04.2015 − Исходная версия.
 */

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using Microsoft.Win32;
using System.IO.Ports;

using AForge.Video.VFW;
using AForge.Video;
using AForge.Video.DirectShow;
using AForge.Controls;

namespace TestCsharp
{
  public partial class ФормаОсновная : Form
  {
    private Усредняющий_Фильтр Фильтр1 = new Усредняющий_Фильтр(80);
    private Усредняющий_Фильтр Фильтр2 = new Усредняющий_Фильтр(24);
    private Усредняющий_Фильтр Фильтр3 = new Усредняющий_Фильтр(24);
    private Усредняющий_Фильтр ФильтрТремора1 = new Усредняющий_Фильтр(2);

    private Системная_Дата_и_Время Время = new Системная_Дата_и_Время();

    private bool КлючУсредняющегоФильтра = true;
    private double ТекущееУсредненноеЗначениеДанныхАЦП = 0;
    private int КлючУсредняющегоФильтраСчетчик = 0;
    private bool КлючДосрочногоВыключенияТаймера = false;

    private double ПредобработанныеДанныеАЦП = 0;

    public ФормаОсновная()
    {
      InitializeComponent();
    }

    /*****************************************************************************************
     * Прототип функции: private void КнопкаЖурналирование_Click(object, EventArgs);
     * Назначение: Функция обработчик события нажатия на кнопку "Журналирование".
     * Аргументы: "sender" − объект отправитель события. "e" − аргументы события.
     * Возвращаемые значения: Нет.
     * Описание: Функция обработчик события нажатия на кнопку "Журналирование".
     *****************************************************************************************/
    private void КнопкаЖурналирование_Click(object sender, EventArgs e)
    {
      if (Журналирование.СтатусИнициализирования == false)
```

```
  {
    Журналирование.Инициализация(this);
  }
  else
  {
    Журналирование.Деинициализация(this);
  }
  График.Очистить(this);
}

/*****************************************************************************************
 * Прототип функции: private void КнопкаВыход_Click(object, EventArgs);
 * Назначение: Функция обработчик события нажатия на кнопку "Выход".
 * Аргументы: "sender" − объект отправитель события. "e" − аргументы события.
 * Возвращаемые значения: Нет.
 * Описание: Функция обработчик события нажатия на кнопку окна "Выход".
 *****************************************************************************************/
private void КнопкаВыход_Click(object sender, EventArgs e)
{
  ВыходИзПрограммы();
}

/*****************************************************************************************
 * Прототип функции: public void ВыходИзПрограммы();
 * Назначение: Функция обработчик события выхода из программы.
 * Аргументы: Нет.
 * Возвращаемые значения: Нет.
 * Описание: Функция обработчик события выхода из программы.
 *****************************************************************************************/
public void ВыходИзПрограммы()
{
  Журналирование.Деинициализация(this);
  Порт.Выключить(this);
  Application.Exit();
}

/*****************************************************************************************
* Прототип функции: private void ФормаОсновная_Load(object, EventArgs);
* Назначение: Функция обработчик события загрузки основной формы.
* Аргументы: "sender" − объект отправитель события. "e" − аргументы события.
* Возвращаемые значения: Нет.
* Описание: Функция обработчик события загрузки основной формы.
*****************************************************************************************/
private void ФормаОсновная_Load(object sender, EventArgs e)
{
  if (SerialPort.GetPortNames().Length != 0)
  {
    Порт.НаличиеДоступныхПортовВсистеме = true;
    СписокПортов.Items.AddRange(SerialPort.GetPortNames());
    СписокПортов.SelectedIndex = 0;
  }
}

/*****************************************************************************************
* Прототип функции: private void serialPort1_DataReceived(object, SerialDataReceivedEventArgs);
* Назначение: Функция обработчик события получения байта по последовательному порту.
* Аргументы: "sender" − объект отправитель события. "e" − аргументы события.
* Возвращаемые значения: Нет.
* Описание: Функция обработчик события получения байта по последовательному порту.
*****************************************************************************************/
private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
  if (Порт.ОбработчикПринятыхБайт(this))
```

```
  {
    string Счетчик = Порт.ПринятыеДанныеСчетчик.ToString();
    double Данные = Порт.ПринятыеДанные;
    Журналирование.ДобавитьДанныеПотока(this, Счетчик, Данные);
    try
    {
      if (this.InvokeRequired)
      {
        BeginInvoke(new MethodInvoker(delegate
        {
          ОсновнойПрограммныйМодуль(Счетчик, Данные);
        }));
      }
    }
    catch
    {
      ОсновнойПрограммныйМодуль(Счетчик, Данные);
    }
  }
}

/*****************************************************************************************
* Прототип функции: private void КнопкаПодключения_Click(object, EventArgs);
* Назначение: Функция обработчик события нажатия на кнопку "Подключение".
* Аргументы: "sender" − объект отправитель события. "e" − аргументы события.
* Возвращаемые значения: Нет.
* Описание: Функция обработчик события нажатия на кнопку "Подключение".
*****************************************************************************************/
private void КнопкаПодключения_Click(object sender, EventArgs e)
{
  if (Порт.СтатусПодключенияКДатчику == false)
  {
    Порт.Включить(this);
  }
  else
  {
    if (Журналирование.СтатусИнициализирования == true)
    {
      Журналирование.Деинициализация(this);
    }
    Порт.Выключить(this);
    График.Очистить(this);
    КнопкаЖурналирование.Enabled = false;
  }
}

/*****************************************************************************************
* Прототип функции: internal void НовыйКадр(object, NewFrameEventArgs);
* Назначение: Функция обработчик события получения нового кадра с видеокамеры.
* Аргументы: "sender" − объект отправитель события. "eventArgs" − аргументы события.
* Возвращаемые значения: Нет.
* Описание: Функция обработчик события получения нового кадра с видеокамеры.
*****************************************************************************************/
internal void НовыйКадр(object sender, NewFrameEventArgs eventArgs)
{
  Журналирование.ДобавитьКадр(this, (Bitmap)eventArgs.Frame.Clone());
}

/*****************************************************************************************
* Прототип функции: private void ФормаОсновная_FormClosing(object, FormClosingEventArgs);
* Назначение: Функция обработчик события закрытия основной формы.
* Аргументы: "sender" − объект отправитель события. "e" − аргументы события.
* Возвращаемые значения: Нет.
```

```
* Описание: Функция обработчик события закрытия основной формы.
**********************************************************************************************/
private void ФормаОсновная_FormClosing(object sender, FormClosingEventArgs e)
{
  ВыходИзПрограммы();
}

/**********************************************************************************************
* Прототип функции: private void ОсновнойПрограммныйМодуль(string,double);
* Назначение: Функция основного модуля идентификации явлений происходящих с хвостом животного.
* Аргументы: "СчетчикПакетов" − счетчик принятых пакетов. "ПринятыеДанныеАЦП" − значение полученное с
ТЧЭ.
* Возвращаемые значения: Нет.
* Описание: Функция основного модуля идентификации явлений происходящих с хвостом животного.
**********************************************************************************************/
private void ОсновнойПрограммныйМодуль(string СчетчикПакетов,double ПринятыеДанныеАЦП)
{
  double Тремор;
  if (КлючУсредняющегоФильтра == true)
  {
    ТекущееУсредненноеЗначениеДанныхАЦП = Фильтр1.ДобавитьДанные(ПринятыеДанныеАЦП);
    График.ПороговыйУровень = 0.5 * ТекущееУсредненноеЗначениеДанныхАЦП;
    Тремор = ФильтрТремора1.ДобавитьДанные(ПринятыеДанныеАЦП) * 0.0;

  }
  else
  {
    Тремор = (ПринятыеДанныеАЦП − ФильтрТремора1.ДобавитьДанные(ПринятыеДанныеАЦП)) * 1.0;
  }
  ПредобработанныеДанныеАЦП = ПринятыеДанныеАЦП − ТекущееУсредненноеЗначениеДанныхАЦП;
  ПредобработанныеДанныеАЦП = Фильтр2.ДобавитьДанные(ПредобработанныеДанныеАЦП);
  ПредобработанныеДанныеАЦП = Фильтр3.ДобавитьДанные(ПредобработанныеДанныеАЦП);
  ПредобработанныеДанныеАЦП += Тремор;

  if (КлючУсредняющегоФильтраСчетчик < Фильтр1.buffer.Length)
  {
    КлючУсредняющегоФильтраСчетчик++;
  }
  else
  {
    КнопкаЖурналирование.Enabled = true;
  }

  if ((ПредобработанныеДанныеАЦП > График.ПороговыйУровень) && (КлючУсредняющегоФильтраСчетчик
== Фильтр1.buffer.Length))
  {
    КлючУсредняющегоФильтра = false;
    Журналирование.Запуск(this);
  }
  else
  {
    КлючУсредняющегоФильтра = true;

    if (КлючДосрочногоВыключенияТаймера == true)
    {
      КлючДосрочногоВыключенияТаймера = false;
      Журналирование.Остановка(this);
    }
  }
  Журналирование.ДобавитьДанные(this, СчетчикПакетов, ПредобработанныеДанныеАЦП);
  График.ДобавитьТочку(this, ПредобработанныеДанныеАЦП);
}
```

```
/*****************************************************************************************
* Прототип функции: private void ВидеорегистрацияТаймер_Tick(object, EventArgs);
* Назначение: Функция обработчик события тика таймера.
* Аргументы: "sender" − объект отправитель события. "e" − аргументы события.
* Возвращаемые значения: Нет.
* Описание: Функция обработчик события тика таймера.
*****************************************************************************************/
private void ВидеорегистрацияТаймер_Tick(object sender, EventArgs e)
{
  КлючДосрочногоВыключенияТаймера = true;
  if (ПредобработанныеДанныеАЦП < График.ПороговыйУровень)
  {
    Журналирование.Остановка(this);
  }
}

 }

}
```

Листинг файла График.cs

```
/*
 * Назначение: Модуль файла графика.
 * Имя файла: График.cs
 * Автор: Сидоренко Михаил Сергеевич
 * Создан: 14.04.2015
 * Список изменений: 14.04.2015 − Исходная версия.
 */

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace TestCsharp
{
  public class График : ФормаОсновная
  {
    static private int Counter_Axis_X = 0;
    static public double ПороговыйУровень = 0;

    /*****************************************************************************************
     * Прототип функции: static public bool ДобавитьТочку(ФормаОсновная, double);
     * Назначение: Функция добавления точки на график.
     * Аргументы: "ЗначениеСДатчика" − значение точки по оси Y.
     * Возвращаемые значения: "true" − достигнуто максимальное число выборок по оси X.
     * "false" − не достигнуто максимальное число выборок по оси X.
     * Описание: Функция добавления точки на график (по оси Y).
     *****************************************************************************************/
    static public bool ДобавитьТочку(ФормаОсновная form, double ЗначениеСДатчика)
    {
      bool state = false;
      form.Chart.Series["Пороговый уровень"].Points.AddXY(Counter_Axis_X, ПороговыйУровень);
      form.Chart.Series["Датчик № 1"].Points.AddXY(Counter_Axis_X, ЗначениеСДатчика);
      Counter_Axis_X++;
      form.ЗначНаГрафТек.Text = "Текущее значение: " + Math.Round(ЗначениеСДатчика,6);
      if (Counter_Axis_X > form.Chart.ChartAreas[0].AxisX.Maximum)
      {
        Очистить(form);
        state = true;
      }
      return state;
    }

    /*****************************************************************************************
    * Прототип функции: static public void Очистить(ФормаОсновная);
    * Назначение: Функция очистки графика.
    * Аргументы: Нет.
    * Возвращаемые значения: Нет.
    * Описание: Функция очистки графика.
    *****************************************************************************************/
    static public void Очистить(ФормаОсновная form)
    {
      form.Chart.Series["Датчик № 1"].Points.Clear();
      form.Chart.Series["Пороговый уровень"].Points.Clear();
      Counter_Axis_X = 0;
      form.ЗначНаГрафТек.Text = "Текущее значение:";
    }

  }
}
```

Листинг файла ЖурналированиеИВидеорегистрация.cs

```
/*
 * Назначение: Модуль файла журналирования и видеорегистрации.
 * Имя файла: ЖурналированиеИВидеорегистрация.cs
 * Автор: Сидоренко Михаил Сергеевич
 * Создан: 15.04.2015
 * Список изменений: 15.04.2015 − Исходная версия.
 */

using System;
using System.Drawing;
using System.Windows.Forms;

using System.IO;

using AForge.Video.VFW;
using AForge.Video;
using AForge.Video.DirectShow;
using AForge.Controls;

namespace TestCsharp
{
  public class Журналирование : ФормаОсновная
  {
    private enum ТипЖурналирования { CSV, AVI };
    static private string Path="";
    static private StreamWriter ФайлЖурналаCSV = null;
    static private StreamWriter ПотокЖурналаCSV = null;
    static private VideoCaptureDevice ИсточникВидеоданных = null;
    static private AVIWriter ПотокЗаписиВидеоданных = null;
    static private int Индекс = 0;

    static public bool СтатусИнициализирования = false;
    static public bool СтатусЖурналирования = false;

    /*****************************************************************************************
     * Прототип функции: static private string ИндексированныйПуть(ТипЖурналирования);
     * Назначение: Функция создания индексированного пути к файлу.
     * Аргументы: "type" − тип журналирования.
     * Возвращаемые значения: Путь к файлу с индексом.
     * Описание: Функция создания индексированного пути к файлу.
     *****************************************************************************************/
    static private string ИндексированныйПуть(ТипЖурналирования type)
    {
      string tmp = null;
      if (Path != "")
      {
        tmp = Path + "_" + Индекс.ToString();
        if (type == ТипЖурналирования.CSV)
        {
          tmp += ".CSV";
        }
        else
        {
          tmp += ".AVI";
        }
      }
      return tmp;
    }

    /*****************************************************************************************
     * Прототип функции: static public bool Инициализация(ФормаОсновная);
```

```
 * Назначение: Функция инициализации журналирования.
 * Аргументы: "form" − основная форма.
 * Возвращаемые значения: Результат выполнения операции.
 * Описание: Функция инициализации журналирования.
 ***********************************************************************************************/
static public bool Инициализация(ФормаОсновная form)
{
  bool tmp = false;
  if (СтатусИнициализирования == false)
  {
    if (form.ДиалоговоеОкноЖурналирование.ShowDialog() == DialogResult.OK)
    {
      Path = form.ДиалоговоеОкноЖурналирование.FileName.ToString();
      if (Path != "")
      {
        VideoCaptureDeviceForm Форма = new VideoCaptureDeviceForm();
        if (Форма.ShowDialog(form) == DialogResult.OK)
        {
          form.МеткаПутиКФайлуЖурнала.Text = "Путь: " + Path;
          form.МеткаСостоянияЖурналирования.Text = "Статус: Журналируется";
          form.КнопкаЖурналирование.Text = "Остановить";
          ИсточникВидеоданных = Форма.VideoDevice;
          form.videoSourcePlayer.VideoSource = ИсточникВидеоданных;
          ИсточникВидеоданных.NewFrame += new NewFrameEventHandler(form.НовыйКадр);
          ПотокЗаписиВидеоданных = new AVIWriter("DIB ");
          ИсточникВидеоданных.Start();
          while (!ИсточникВидеоданных.IsRunning);
          ПотокЖурналаCSV = File.CreateText(Path+"ВесьПоток.CSV");
          СтатусИнициализирования = true;
        }
      }
    }
  }
  return tmp;
}

/***********************************************************************************************
 * Прототип функции: static public bool Деинициализация(ФормаОсновная);
 * Назначение: Функция деинициализации журналирования.
 * Аргументы: "form" − основная форма.
 * Возвращаемые значения: Результат выполнения операции.
 * Описание: Функция деинициализации журналирования.
 ***********************************************************************************************/
static public bool Деинициализация(ФормаОсновная form)
{
  bool tmp = false;
  if (СтатусИнициализирования == true)
  {
    if (СтатусЖурналирования == true)
    {
      tmp = Остановка(form);
    }
    ФайлЖурналаCSV = null;
    ИсточникВидеоданных.SignalToStop();
    while(ИсточникВидеоданных.IsRunning);
    ИсточникВидеоданных = null;
    ПотокЗаписиВидеоданных = null;
    form.МеткаПутиКФайлуЖурнала.Text = "Путь:";
    form.МеткаСостоянияЖурналирования.Text = "Статус: Не журналируется";
    form.КнопкаЖурналирование.Text = "Запустить";
    Path = "";
    Индекс = 0;
    ПотокЖурналаCSV.Close();
```

```
      ПотокЖурналаCSV = null;
      СтатусИнициализирования = false;
      tmp = true;
    }
    return tmp;
  }

  /*****************************************************************************************
   * Прототип функции: static public bool Запуск(ФормаОсновная);
   * Назначение: Функция запуска журналирования.
   * Аргументы: "form" − основная форма.
   * Возвращаемые значения: Результат выполнения операции.
   * Описание: Функция запуска журналирования.
   *****************************************************************************************/
  static public bool Запуск(ФормаОсновная form)
  {
    bool tmp = false;
    if ((СтатусИнициализирования==true) && (СтатусЖурналирования == false))
    {
      ФайлЖурналаCSV = File.CreateText(ИндексированныйПуть(ТипЖурналирования.CSV));
      ПотокЗаписиВидеоданных.Open(ИндексированныйПуть(ТипЖурналирования.AVI),
ИсточникВидеоданных.VideoResolution.FrameSize.Width,
ИсточникВидеоданных.VideoResolution.FrameSize.Height);
      form.ВидеорегистрацияТаймер.Enabled = true;
      СтатусЖурналирования = true;
    }
    return tmp;
  }

  /*****************************************************************************************
   * Прототип функции: static public bool Остановка(ФормаОсновная);
   * Назначение: Функция остановки журналирования.
   * Аргументы: "form" − основная форма.
   * Возвращаемые значения: Результат выполнения операции.
   * Описание: Функция остановки журналирования.
   *****************************************************************************************/
  static public bool Остановка(ФормаОсновная form)
  {
    bool tmp = false;
    if ((СтатусИнициализирования == true) && (СтатусЖурналирования == true))
    {
      ФайлЖурналаCSV.Close();
      ПотокЗаписиВидеоданных.Close();
      form.ВидеорегистрацияТаймер.Enabled = false;
      Индекс++;
      СтатусЖурналирования = false;
    }
    return tmp;
  }

  /*****************************************************************************************
   * Прототип функции: static public void ДобавитьКадр(ФормаОсновная, Bitmap);
   * Назначение: Функция добавления кадра в видеофайл.
   * Аргументы: "form" − основная форма. "Кадр" − кадр с видеокамеры.
   * Возвращаемые значения: Нет.
   * Описание: Функция добавления кадра в видеофайл.
   *****************************************************************************************/
  static public void ДобавитьКадр(ФормаОсновная form, Bitmap Кадр)
  {
    if ((СтатусИнициализирования == true) && (СтатусЖурналирования == true))
    {
      try
      {
```

```
        ПотокЗаписиВидеоданных.AddFrame(Кадр);
      }
      catch
      {
        Кадр = null;
      }
    }
  }

  /*****************************************************************************************
   * Прототип функции: static public void ДобавитьДанные(ФормаОсновная, string, double);
   * Назначение: Функция добавления кадра в видеофайл.
   * Аргументы: "form" − основная форма. "время" − текущее системное время. "Данные" − данные с ТЧЭ.
   * Возвращаемые значения: Нет.
   * Описание: Функция добавления кадра в видеофайл.
   *****************************************************************************************/
  static public void ДобавитьДанные(ФормаОсновная form, string время, double Данные)
  {
    if ((СтатусИнициализирования == true) && (СтатусЖурналирования == true))
    {
      ФайлЖурналаCSV.WriteLine(время + ";" + Данные.ToString());
    }
  }

  /*****************************************************************************************
   * Прототип функции: static public void ДобавитьДанныеПотока(ФормаОсновная, string, double);
   * Назначение: Функция добавления новых данных с ТЧЭ в файл.
   * Аргументы: "form" − основная форма. "время" − текущее системное время. "Данные" − данные с ТЧЭ.
   * Возвращаемые значения: Нет.
   * Описание: Функция добавления новых данных с ТЧЭ в файл.
   *****************************************************************************************/
  static public void ДобавитьДанныеПотока(ФормаОсновная form, string время, double Данные)
  {
    if (СтатусИнициализирования == true)
    {
      ПотокЖурналаCSV.WriteLine(время + ";" + Данные.ToString());
    }
  }

 }
}
```

Листинг файла Порт.cs

```
/*
 * Назначение: Модуль файла последовательного порта.
 * Имя файла: Порт.cs
 * Автор: Сидоренко Михаил Сергеевич
 * Создан: 14.04.2015
 * Список изменений: 14.04.2015 − Исходная версия.
 */

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace TestCsharp
{
  public class Порт : ФормаОсновная
  {
    static private int PackageByteCounter = 0;
    static private int PayloadByteCounter = 0;
    static private byte[] RecivBuffer = new byte[8];
    static public double ПринятыеДанные = 0;
    static public double ПринятыеДанныеСчетчик = 0;
    static public bool НаличиеДоступныхПортовВсистеме = false;
    static public bool СтатусПодключенияКДатчику = false;

    /*****************************************************************************************
     * Прототип функции: static private double ПарсерПоляСопротивление(byte);
     * Назначение: Функция парсинга поля сопротиления.
     * Аргументы: "data" − индекс сопротивления.
     * Возвращаемые значения: Сопротивление в кОм.
     * Описание: Функция парсинга поля сопротиления.
     *****************************************************************************************/
    static private double ПарсерПоляСопротивление(byte data)
    {
      double tmp;
      switch (data)
      {
        case 1:
        {
          tmp = 82;
          break;
        }
        case 2:
        {
          tmp = 39;
          break;
        }
        case 3:
        {
          tmp = 26.43;
          break;
        }
        case 4:
        {
          tmp = 22;
          break;
        }
        case 5:
        {
          tmp = 17.35;
          break;
```

```
    }
    case 6:
    {
      tmp = 14.07;
      break;
    }
    case 7:
    {
      tmp = 12.01;
      break;
    }
    case 8:
    {
      tmp = 10;
      break;
    }
    case 9:
    {
      tmp = 8.91;
      break;
    }
    case 10:
    {
      tmp = 7.96;
      break;
    }
    case 11:
    {
      tmp = 7.25;
      break;
    }
    case 12:
    {
      tmp = 6.88;
      break;
    }
    case 13:
    {
      tmp = 6.34;
      break;
    }
    case 14:
    {
      tmp = 5.84;
      break;
    }
    case 15:
    {
      tmp = 5.46;
      break;
    }
    default:
    {
      tmp = 0;
      break;
    }
  }
  return tmp;
}

/*****************************************************************************************
 * Прототип функции: static private void ОчиститьБуферПринятыхДанных(ФормаОсновная);
 * Назначение: Функция очистки буфера принятых данных последовательного порта.
```

```
 * Аргументы: "form" − основная форма.
 * Возвращаемые значения: Нет.
 * Описание: Функция очистки буфера принятых данных последовательного порта.
 ****************************************************************************************************/
static private void ОчиститьБуферПринятыхДанных(ФормаОсновная form)
{
  Array.Clear(RecivBuffer, 0, RecivBuffer.Length);
  PackageByteCounter = 0;
  PayloadByteCounter = 0;
  if (form.SerialPort.IsOpen == true)
  {
    form.SerialPort.DiscardInBuffer();
    form.SerialPort.DiscardOutBuffer();
  }
}

/****************************************************************************************************
 * Прототип функции: static public bool ОбработчикПринятыхБайт(ФормаОсновная);
 * Назначение: Функция обработчика принятых байт по последовательному порту.
 * Аргументы: "form" − основная форма.
 * Возвращаемые значения: Признак завершения обработки пакета данных.
 * "true" − получен новый пакет данных. "false" − нет нового пакета данных.
 * Описание: Функция обработчика принятых байт по последовательному порту.
 ****************************************************************************************************/
static public bool ОбработчикПринятыхБайт(ФормаОсновная form)
{
  bool state = false;
  byte tmp = 0;
  if (form.SerialPort.IsOpen == true)
  {
    tmp = Convert.ToByte(form.SerialPort.ReadByte());
  }
  switch (PackageByteCounter)
  {
    case 0:
      {
        if (tmp == 0xAA)
        {
          RecivBuffer[PackageByteCounter++] = tmp;
        }
        else
        {
          ОчиститьБуферПринятыхДанных(form);
        }
        break;
      }
    case 1:
      {
        if ((tmp == 0xA5) || (tmp == 0x5A))
        {
          RecivBuffer[PackageByteCounter++] = tmp;
        }
        else
        {
          ОчиститьБуферПринятыхДанных(form);
        }
        break;
      }
    case 2:
      {
        RecivBuffer[PayloadByteCounter + 2] = tmp;
        PayloadByteCounter++;
        if (PayloadByteCounter == (4 + 2))
```

```
        {
          if ((RecivBuffer[0] == 0xAA) && ((RecivBuffer[1] == 0x5A) || (RecivBuffer[1] == 0xA5)) &&
(RecivBuffer[RecivBuffer.Length − 1] == 0X55))
          {
            Контрольная_сумма_XOR XORSum = new Контрольная_сумма_XOR();
            byte[] PayloadBuffer = new byte[4];
            PayloadBuffer[0] = RecivBuffer[2];
            PayloadBuffer[1] = RecivBuffer[3];
            PayloadBuffer[2] = RecivBuffer[4];
            PayloadBuffer[3] = RecivBuffer[5];
            if (RecivBuffer[RecivBuffer.Length − 2] == XORSum.GetXOR(PayloadBuffer, PayloadBuffer.Length))
            {
              double ПринятыеДанныеАЦП = Convert.ToDouble(BitConverter.ToUInt16(PayloadBuffer, 0));
              double ПринятыеДанныеСопротивления = ПарсерПоляСопротивление(PayloadBuffer[2]);
              ПринятыеДанныеСчетчик = Convert.ToDouble(PayloadBuffer[3]);
              ПринятыеДанные = (ПринятыеДанныеАЦП * ПринятыеДанныеСопротивления) / (1024.0 −
ПринятыеДанныеАЦП);
              state = true;
            }
          }
          ОчиститьБуферПринятыхДанных(form);
        }
        break;
      }
    default:
      {
        ОчиститьБуферПринятыхДанных(form);
        break;
      }
  }
  return state;
}

/*****************************************************************************************
 * Прототип функции: static public void Выключить(ФормаОсновная);
 * Назначение: Функция включения последовательного порта и обработчика принятых байт.
 * Аргументы: "form" − основная форма.
 * Возвращаемые значения: Нет.
 * Описание: Функция включения последовательного порта и обработчика принятых байт.
 *****************************************************************************************/
static public void Выключить(ФормаОсновная form)
{
  if (СтатусПодключенияКДатчику == true)
  {
    ОчиститьБуферПринятыхДанных(form);
    ПринятыеДанные = 0;
    form.СписокПортов.Enabled = true;
    form.МеткаСостоянияПодключенияКДатчику.Text = "Статус: Не подключен";
    form.КнопкаПодключения.Text = "Подключить";
    form.SerialPort.Close();
    СтатусПодключенияКДатчику = false;
  }
}

/*****************************************************************************************
 * Прототип функции: static public void Включить(ФормаОсновная);
 * Назначение: Функция выключения последовательного порта и обработчика принятых байт.
 * Аргументы: "form" − основная форма.
 * Возвращаемые значения: Нет.
 * Описание: Функция выключения последовательного порта и обработчика принятых байт.
 *****************************************************************************************/
static public void Включить(ФормаОсновная form)
{
```

```
    if ((НаличиеДоступныхПортовВсистеме == true) && (СтатусПодключенияКДатчику == false))
    {
      form.СписокПортов.Enabled = false;
      form.МеткаСостояниеПодключенияКДатчику.Text = "Статус: Подключен";
      form.КнопкаПодключения.Text = "Отключить";
      form.SerialPort.PortName = form.СписокПортов.SelectedItem.ToString();
      form.SerialPort.Open();
      СтатусПодключенияКДатчику = true;
    }
  }

 }

}
```

Листинг файла СервисныеКлассы.cs

```
/*
 * Назначение: Модуль файла сервисных классов и функций.
 * Имя файла: СервисныеКлассы.cs
 * Автор: Сидоренко Михаил Сергеевич
 * Создан: 18.03.2015
 * Список изменений: 18.03.2015 − Исходная версия.
 */

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace TestCsharp
{
  public class RandomValue
  {
    private Random random = new Random();
    private double tmp = 0.0;

    /*****************************************************************************************
    * Прототип функции: public UInt16 GetRandomvalue();
    * Назначение: Функция генерации случайного числа типа "UInt16".
    * Аргументы: Нет.
    * Возвращаемые значения: Число типа "UInt16".
    * Описание: Функция генерации случайного числа типа "UInt16" из диапазона от 0 до 1023.
*****************************************************************************************/
public UInt16 GetRandomvalue()
    {
      tmp = (1023.0 * (random.Next(0, 100) / 100.0));
      return Convert.ToUInt16(tmp);
    }

        /*****************************************************************************************
    * Прототип функции: public UInt16 GetRandomvalue(UInt16, UInt16);
    * Назначение: Функция генерации случайного числа типа "UInt16".
    * Аргументы: "lowerbound" − нижняя граница. "upperbound" − верхняя граница.
    * Возвращаемые значения: Число типа "UInt16".
    * Описание: Функция генерации случайного числа типа "UInt16" из диапазона от 0 до 1023.
*****************************************************************************************/
public UInt16 GetRandomvalue(UInt16 lowerbound, UInt16 upperbound)
    {
      tmp = ((Convert.ToDouble(upperbound) − Convert.ToDouble(lowerbound)) * (random.Next(0, 100) / 100.0) +
Convert.ToDouble(lowerbound));
      return Convert.ToUInt16(tmp);
    }

    /*****************************************************************************************
     * Прототип функции: public double GetRandomvalue(int);
     * Назначение: Функция генерации случайного числа.
     * Аргументы: "data" − опорное значение для генерирования числа.
     * Возвращаемые значения: Сгенерированное число.
     * Описание: Функция генерации случайного числа в диапазоне от 0.95*data до 1.05*data.
     *****************************************************************************************/
    public double GetRandomvalue(int data)
    {
      double upperbound = Convert.ToDouble(data)*1.05;
      double lowerbound = Convert.ToDouble(data)*0.95;
      return ((upperbound − lowerbound) * (random.Next(0, 100) / 100.0) + lowerbound);
    }
```

```
}

public class Усредняющий_Фильтр
{
  public double[] buffer;
  private int counter;
  private double Sum;

  /*****************************************************************************************
   * Прототип функции: public Усредняющий_Фильтр(int);
   * Назначение: Конструктор класса Усредняющий_Фильтр.
   * Аргументы: "buffer_length" − длина буфера.
   * Описание: Конструктор класса Усредняющий_Фильтр.
   *****************************************************************************************/
  public Усредняющий_Фильтр(int buffer_length)
  {
    buffer = new double[buffer_length];
    counter=0;
    Sum=0;
  }

  /*****************************************************************************************
   * Прототип функции: public double ДобавитьДанные(double);
   * Назначение: Функция добавления данных в усредняющий фильтр.
   * Аргументы: "data" − добовляемое значение в фильтр.
   * Возвращаемые значения: Усредненное значение.
   * Описание: Функция добавления данных в усредняющий фильтр.
   *****************************************************************************************/
  public double ДобавитьДанные(double data)
  {
    Sum −= buffer[counter];
    buffer[counter++] = data;
    Sum += data;
    if (counter == buffer.Length)
    {
      counter = 0;
    }
    return (Sum / buffer.Length);
  }
}

public class Системная_Дата_и_Время
{
  private DateTime Текущяя_Дата_Время;

  /*****************************************************************************************
   * Прототип функции: public string GetCurrentData();
   * Назначение: Функция получения текущей даты.
   * Аргументы: Нет.
   * Возвращаемые значения: Текущая дата.
   * Описание: Функция получения текущей даты.
   *****************************************************************************************/
  public string GetCurrentData()
  {
    Текущяя_Дата_Время = DateTime.Now;
    return (Текущяя_Дата_Время.Day.ToString() + "." + Текущяя_Дата_Время.Month.ToString() + "." +
Текущяя_Дата_Время.Year.ToString());
  }

  /*****************************************************************************************
   * Прототип функции: public string GetCurrentTime();
   * Назначение: Функция получения текущего времени.
   * Аргументы: Нет.
```

```
     * Возвращаемые значения: Текущее время.
     * Описание: Функция получения текущего времени.
     ***********************************************************************************************/
    public string GetCurrentTime()
    {
      Текущяя_Дата_Время = DateTime.Now;
      return (Текущяя_Дата_Время.Hour.ToString() + "." + Текущяя_Дата_Время.Minute.ToString() + "." +
Текущяя_Дата_Время.Second.ToString() + "." + Текущяя_Дата_Время.Millisecond.ToString());
    }

    /***********************************************************************************************
     * Прототип функции: public string GetCurrentDataTime();
     * Назначение: Функция получения текущей даты и времени.
     * Аргументы: Нет.
     * Возвращаемые значения: Текущая дата и время.
     * Описание: Функция получения текущей даты и времени.
     ***********************************************************************************************/
    public string GetCurrentDataTime()
    {
      Текущяя_Дата_Время = DateTime.Now;
      return (Текущяя_Дата_Время.Day.ToString() + "." + Текущяя_Дата_Время.Month.ToString() + "." +
Текущяя_Дата_Время.Year.ToString() + ":" + Текущяя_Дата_Время.Hour.ToString() + "." +
Текущяя_Дата_Время.Minute.ToString() + "." + Текущяя_Дата_Время.Second.ToString() + "." +
Текущяя_Дата_Время.Millisecond.ToString());
    }

    /***********************************************************************************************
     * Прототип функции: public string GetCurrentFormatDate();
     * Назначение: Функция получения формата текущей даты.
     * Аргументы: Нет.
     * Возвращаемые значения: Текущий формат даты.
     * Описание: Функция получения формата текущей даты.
     ***********************************************************************************************/
    public string GetCurrentFormatDate()
    {
      return ("Day.Month.Year");
    }

    /***********************************************************************************************
     * Прототип функции: public string GetCurrentFormatTime();
     * Назначение: Функция получения формата текущего времени.
     * Аргументы: Нет.
     * Возвращаемые значения: Текущий формат времени.
     * Описание: Функция получения формата текущего времени.
     ***********************************************************************************************/
    public string GetCurrentFormatTime()
    {
      return ("Hour.Minute.Second.Millisecond");
    }

    /***********************************************************************************************
     * Прототип функции: public string GetCurrentFormatDataTime();
     * Назначение: Функция получения формата текущей даты и времени.
     * Аргументы: Нет.
     * Возвращаемые значения: Текущий формат даты и времени.
     * Описание: Функция получения формата текущей даты и времени.
     ***********************************************************************************************/
    public string GetCurrentFormatDataTime()
    {
      return (GetCurrentFormatDate() + ":" + GetCurrentFormatTime());
    }
  }
```

```
public class Контрольная_сумма_XOR
{
  /*****************************************************************************************************
   * Прототип функции: public byte GetXOR(byte[], int);
   * Назначение: Функция вычисления контрольной суммы массива байт.
   * Аргументы: "buffer" − буфер по которому вычисляется контрольная сумма. "buffer_length" − длина буфера.
   * Возвращаемые значения: Результат выполнения функции. Контрольная сумма.
   * Описание: Функция вычисления контрольной суммы массива байт.
   *****************************************************************************************************/
  public byte GetXOR(byte[] buffer, int buffer_length)
  {
    byte tmp = buffer[0];
    for (int i = 1; i < buffer_length; i++)
    {
      tmp ^= buffer[i];
    }
    return tmp;
  }
}
}
```