УДК 618.19-006-07:616.073.756.8-027.552

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ МНОГОПРОЦЕССОРНОЙ АРХИТЕКТУРЫ ПЕРСОНАЛЬНОГО ЭЛЕКТРОИМПЕДАНСНОГО МАММОГРАФА ПЭМ

И. К. Лакеев, А. В. Корженевский, Т. С. Туйкин

ИРЭ им. В.А. Котельникова РАН, 125009, Москва, ул. Моховая 11, корп.7

Статья поступила в редакцию 8 декабря 2017 г.

Аннотация. Предложен способ решения проблемы синхронизации нескольких микроконтроллеров с помощью состояния легкого сна и последующего пробуждения в процессе сбора полного набора данных с помощью многопроцессорной архитектуры измерительной системы персонального электроимпедансного маммографа, в виду несостоятельности подхода синхронизации исключительно с помощью прерываний. Показано, что безошибочное использование одного канала UART со стороны управляющего микроконтроллера для обслуживания двух каналов зависимых микропроцессоров возможно в случае конфигурации каналов в режиме только по требованию. Предложен подход к разработке и отладке программного обеспечения, позволяющий существенно сократить затрачиваемое время на получение устойчивого решения за счет использования сред, в основе которых лежит язык программирования python, поддерживающих управление параллельными кластерами, что позволяет отказаться от линейного подхода к разработке. Это было продемонстрировано на примере непоследовательного анализа данных, полученных от измерительной системы, на необходимом уровне абстракции в любой последовательности.

Ключевые слова: программно-аппаратный комплекс, многопроцессорная архитектура, проблемы синхронизации, разработка и отладка, язык программирования python.

Abstract. Optimized software development for symmetric multiprocessing systems, especially the low-level ones, has always been hard in terms of achieving sustainable

operational level with reasonable effort and in acceptable time. Through the process of personal electrical impedance mammograph software development and debugging it was shown that major drawback of Harvard-type architecture, implemented in modern AVR microprocessors, may be an issue in terms of precise interrupt-driven synchronization between several elements of the multiprocessor architecture because of the different length of commands and therefore often unpredictable execution timing before finishing the very last command and entering the particular interruption, that can be successfully mitigated in two steps: initial configuration of the job that needs to be done with respect to timing which should be followed by light sleep state and further synchronization interruption that will be able to wake up the modules and provide exact timing. It was also shown that during parallel usage of one UART of the micro controller unit to serve two dependent units one may experience frame errors unless UART reconfigured on demand. Utilization of any python-based computational environments that support special kernels with enhanced introspection and parallel computing clusters management with the help of asynchronous status callbacks can significantly improve the development process speeding up the debugging routine. It is shown in the example of personal electrical impedance mammograph software development and debugging, especially on the late stages of development when the level of abstraction gets higher, particularly on the examples of experiment data results assessment on multiple levels from raw data to reconstructed images.

Keywords: hardware and software system, multiprocessing system, synchronization problem solving, debugging and development, python.

1. Введение

На рисунке 1 приведена блок-схема архитектуры измерительной системы для сбора данных при исследовании молочных желез методом электро-импедансной томографии, используемая в приборе медицинского назначении маммографе электроимпедансном многочастотном (МЭМ), разработанном в Институте радиотехники и электроники им. В.А.Котельникова РАН (ИРЭ им.

В.А.Котельникова РАН). Особенность метода визуализации, примененного в МЭМ [1, 2], заключается в том, что реализована статическая визуализация, в отличие от метода, описанного в [3, 4], обеспечивающего только динамическую визуализацию.

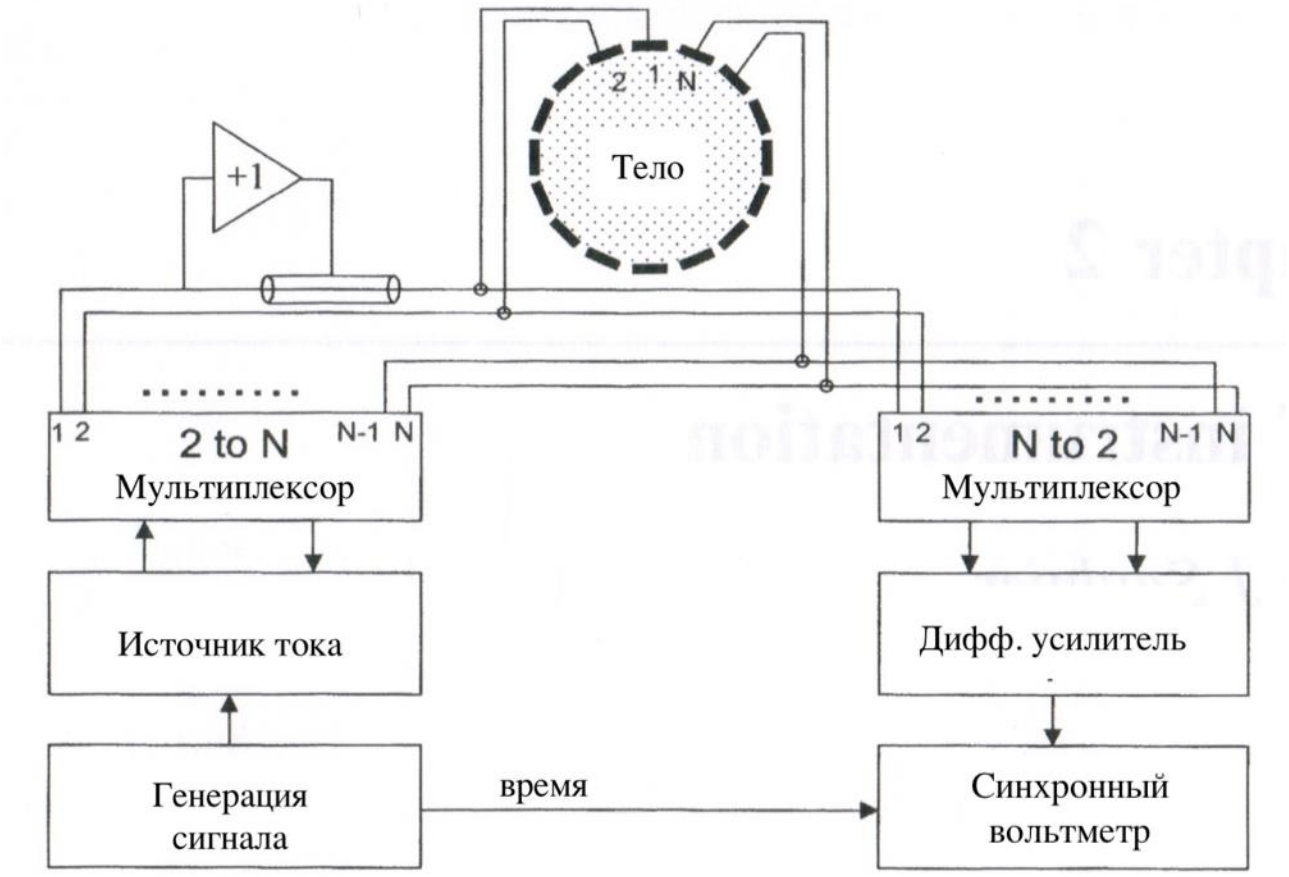

Рисунок 1 – блок-схема измерительной системы ЭИТ

Одной из основных проблем, связанных с существующей системой сбора данных с помощью метода электроимпедансной томографии, при таком подходе является скорость роста количества комбинаций генераторов и приемников с ростом количества электродов в измерительной системе (далее обозначим как n), и как следствие рост длительности одного измерения. Каждое измерение представляет собой при таком подходе инжектирование меандра с помощью электрода-генератора и его синхронное измерение с помощью электрода-приемника, для простоты расчетов зафиксируем частоту

этого меандра на уровне $f = 50$ кГц и количество периодов, подлежащих измерению, на уровне $d = 10$. Таким образом оценка снизу времени, необходимого для проведения одного измерения с помощью пары электродов генератора и приемника составляет $t = \frac{1}{f} \cdot d$. Измерительная матрица МЭМ состоит из $n = 256$ электродов, что означает, что для сбора полного, в математическом смысле, набора данных необходимо получить $\frac{n \cdot (n-1)}{2}$ измерений, в силу взаимности измерений и невозможности электрода-генератора быть одновременно электродом-приемником. Время, необходимое для сбора полного набора данных, можно оценить как $T = t \cdot \frac{n \cdot (n-1)}{2} \approx 6{,}5$ секунд. На практике, работа измерительной системы несколько сложнее, включает в себя несколько дополнительных шагов, которые требуют времени исполнения во время работы микропроцессора: работу алгоритма, способного скомпенсировать в том числе гальванические эффекты между каждой парой генератор-приемник для корректности каждого измерения, составление карты контактов, удовлетворяющих условиям для того, чтобы считать полученные измерения корректными, и другие так называемые транзакционные издержки по времени, связанные с работой всей измерительной системы в целом. В случае прибора МЭМ реальное время одного измерения $T \approx 30$ секунд. В условиях проведения реальных исследований такой временной отрезок несет в себе достаточно серьезные риски для достоверности каждого исследования, и одним из направлений работ стало исследование возможности ускорения сбора полного набора данных.

В ходе исследования разрабатывался новый прибор «персональный электроимпедансный маммограф (ПЭМ)» со следующими двумя принципиальными отличиями в архитектуре измерительной системы:

- меньшее количество электродов в измерительной матрице $n = 120$,
- использование многопроцессорной архитектуры для обеспечения параллельности измерений, что позволит ускорить скорость сбора полного набора данных.

2. Описание архитектурного решения многопроцессорной архитектуры

В ходе исследований было принято решение использовать в качестве измерительных микроконтроллеров 8 х ATMEL XMEGA32E5 и один ATMEL XMEGA256A3U в качестве управляющего микропроцессора, ответственного за управление всей логикой измерительной системы; а также модуль Bluetooth HC-05 для работы в роли модуля связи с персональным компьютером для приема и обработки команд, передачи данных. Выбор в пользу семейства XMEGA обусловлен сравнительно низкой ценой, одновременным наличием АЦП на борту каждого элемента с программно-управляемым аналоговым усилением и другими характеристиками, выбор HC-05 обусловлен простотой конфигурации и работы при одновременно доступной цене.

Ниже на рисунке 2 можно увидеть схему реализации подключения 8 измерительных микроконтроллеров XMEGA32E5, к каждому из которых подключены 15 электродов из измерительной матрицы, состоящей из 120 измерительных электродов.

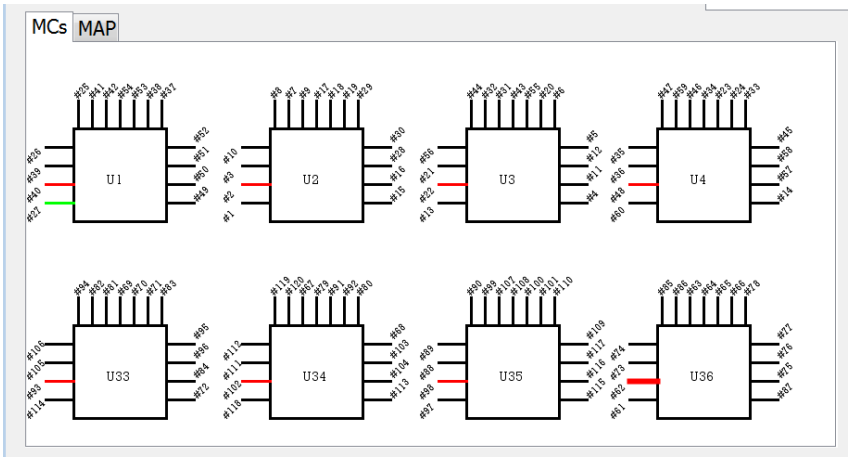

Рисунок 2 – визуализация измерительной конфигурации ПЭМ в отладочном программном обеспечении первой версии

На рисунке 2 приведен пример одной из измерительных конфигураций, где зеленым отмечен электрод-генератор, красным отмечены электроды-приемники, которые должны синхронно измерять инжектируемый меандр.

При создании программного обеспечения был выбран следующий подход:

- Для XMEGA32E5 создается одно программное обеспечение, которое будет идентичным для всех 8 микроконтроллеров, далее для простоты они будут обозначены словом SLAVE.
- Программное обеспечение XMEGA256A3U является управляющим, в дальнейшем будет обозначено словом MASTER, основными задачами которого являются:
    - Управление логикой измерительных микроконтроллеров SLAVE с помощью команд через выделенные UART.
    - Управление логикой измерения и перебором измерительных конфигураций.
    - Обеспечение синхронности измерений за счет отдельного канала синхронизации SYNC, доступного для всех SLAVE одновременно.
    - Обеспечение синхронного измерения протекшего тока в измерительной системе с помощью АЦП и измерительной цепи, в которую включены два отводящих электрода.
    - Сбор, обработка и передача полученных измерений.

3. Основные трудности и пути их преодоления

В ходе работы можно выделить несколько ключевых проблем, которые потребовали существенного количества времени для решения:

1. Обеспечение синхронизации измерений: несмотря на наличие канала SYNC, проверка синхронности показывала расхождение во временах работы SLAVE-ов на несколько тактов ведущей частоты.
2. Обеспечение высокой скорости передачи данных между MASTER и SLAVE в первой версии программного обеспечения и текущем состоянии архитектурного решения, когда на одном из 4 UART каналах находятся два SLAVE микроконтроллера, а адресность сообщения определялась с помощью высокого уровня на соответствующем канале ENABLE от 1 до

8, на скоростях выше 115200 бод/секунду происходили ошибки в кадрах UART.

3. Необходимость следовать последовательной логике при разработке отладочного программного обеспечения, пример которого приведен на рисунке 2, что существенно замедляло процесс разработки из-за:
    - долгой идентификации места ошибок в логике работы измерительной системе (MASTER, или SLAVE),
    - необходимости строго соблюдения протокола передачи данных между всеми тремя программными обеспечениям.

В случае первой проблемы трудность заключалась в том, что только передача фронта по SYNC не обеспечивала нужный уровень синхронизации: в зависимости от того, какие прерывания исполнялись в процессе выполнения команд MASTER, SLAVE-ы находились в разных состояниях даже основного цикла ожидания в силу фундаментальной особенности гарвардской архитектуры, когда разные команды могут иметь разное время исполнения, и в случае обработки прерывания из-за фронта на SYNC отдельные процессоры завершали выполнение команд разной длины, прежде чем обработать прерывание. В результате исследований было найдено следующее стабильное решение.

Перед каждым шагом, на котором требуется прецизионная синхронизация с точностью до 1 такта несущей частоты, необходимо перевести все микроконтроллеры в режим легкого сна, предварительно сконфигурировав их таким образом, чтобы в результате создания фронта на канале SYNC все 8 микропроцессоров выходили из состояния сна и выполняли действия, требующие синхронизации (например, синхронное измерение инжектируемого меандра).

Причина второй трудности заключалась в том, что необходимо было реализовать более сложную логику конфигурации UART каналов на соответствующих SLAVE микроконтроллерах только по прерыванию, вызванному высоким уровнем на ENABLE. Постоянная же конфигурация

UART каналов и лишь очистка буфера в случае нецелевого получения сообщений выполняла свою функцию только при передаче данных из MASTER к другому SLAVE, но не наоборот. В случае передачи данных от одного SLAVE к MASTER-у, например, при выгрузке большого объема данных, происходил конфликт между изменением уровня на канале RX передающего SLAVE-а и поддержкой постоянного уровня на том же канале RX у пассивного SLAVE. Последовательная конфигурация же нужного UART только в случаях адресного сообщения и обработки прерывания на соответствующем канале ENABLE позволила увеличить скорость обмена данными внутри многопроцессорной архитектуры до 921600 бод/секунду.

Третья проблема носит более общий характер. Первая версия отладочного программного обеспечения сначала разрабатывалась на языке JAVA в среде NetBeans IDE [5]. Существенное замедление скорости разработки в основном выражалось в необходимости последовательного внесения изменения в интерфейс, алгоритм работы отладочного программного обеспечения, последующее проведение эксперимента и затем анализа результатов. Визуализация результатов измерений эксперимента внутри отладочного программного обеспечения требовала строгого соблюдения протокола приема и передачи команд, обработки потока данных, что зачастую приводило к существенному усложнению отладки всех трех программных обеспечений и поиску ошибок, прежде чем сам эксперимент был бы проведен. Интерпретация же отдельно сохраненных результатов требовала написания каждый раз мини-программ в сторонних средах разработки или пакетных решениях, что также не приводило к ускорению процесса разработки и отладки.

В ходе исследований разработка отладочного программного обеспечения была переведена в среду Jupyter Notebook на язык Python 2.7 из свободно распространяемого пакета Anaconda [6], обычным предназначением которого принято считать анализ данных в других сферах, например, в задачах машинного обучения. Большим достоинством этого пакета является наличие

оболочки для установки сторонних библиотек из облачного репозитория anaconda-cloud с помощью простой команды conda –install <название пакета из anaconda cloud>, а также возможность асинхронного выполнения команд внутри программы. В ходе разработок использовался следующий подход:

- библиотека Pyserial, доступная на Anaconda-cloud, обеспечивала связь персонального компьютера и модуля связи HC-05 в измерительной системе с помощью протокола, очень похожего на привычный многим serial com port,
- использования отдельного процесса на прием позволяло в постоянном режиме принимать и хранить все данные полученные от измерительной системы,
- возможность «дописывать» и вызывать функции по обработке полученных данных позволяла избавиться от жестких требований соответствий протокола приема и передачи данных за счет возможности написания функций и методов «на ходу».

Сценарий использования такого подхода в результате мог сводиться к следующему:

1. Разрабатывались изменения в программном обеспечении SLAVE, добавлялась возможно считывать уровень АЦП на соответствующих электродах 1-15.
2. Разрабатывались изменения в программном обеспечении MASTER, добавлялась возможность опроса каждого из 8 микропроцессоров и чтение уровня АЦП на соответствующем электроде 1-15.
3. В Jupyter notebook генерировались команды master, полученные данные складывались в строковую переменную, с возможностью на ходу изменения логики работы и проведения эксперимента:
    a. выбор только 1 электрода одного процессора,
    b. выбор только 15 электродов одного процессора,
    c. выбор 15 электродов нескольких выбранных процессоров, и т.д.

4. Полученные данные обрабатывались уже после получения с помощью различных функций и методов, способных ответить на интересующие вопросы, тем самым реализуя весь потенциал возможностей анализа данных в средах, использующих python.

Так, изображения на рисунке 3 соответствуют наглядным визуализациям распределений потенциалов на поверхности при выставлении каждого из соответствующих электродов в состояние генератора.

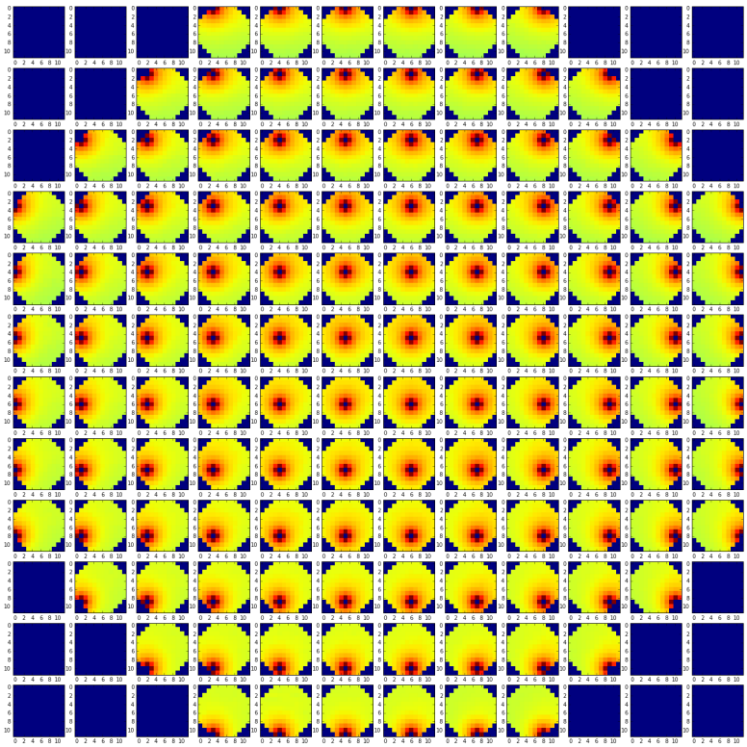

Рисунок 3 – визуализация полного в математическом смысле набора данных, созданная с использованием отладочного программного обеспечения на языке python

Изображения на рисунке 4 представляют собой визуализацию распределений проводимостей на каждом томографическом срезе определенной глубины после реконструкции.

Изображения на рисунке 5 соответствуют визуализации исходных данных уровней АЦП на выбранном процессоре 1 при зафиксированном генераторе 0 для проверки корректности измерений в процессе инжектирования меандра.

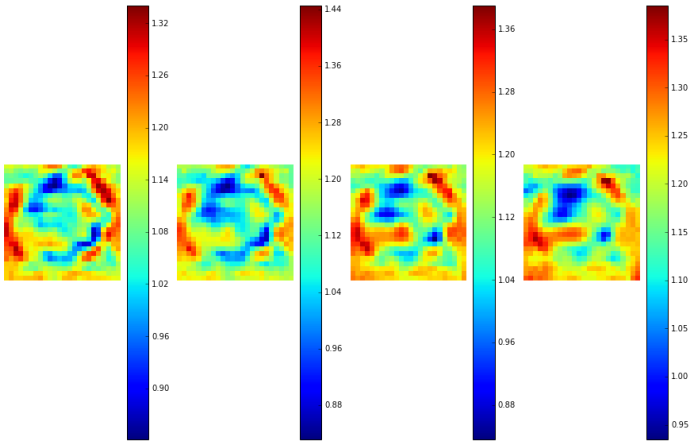

Рисунок 4 – визуализация распределения проводимостей на 4 томографических срезах разной глубины, созданная с помощью отладочного программного обеспечения на языке python

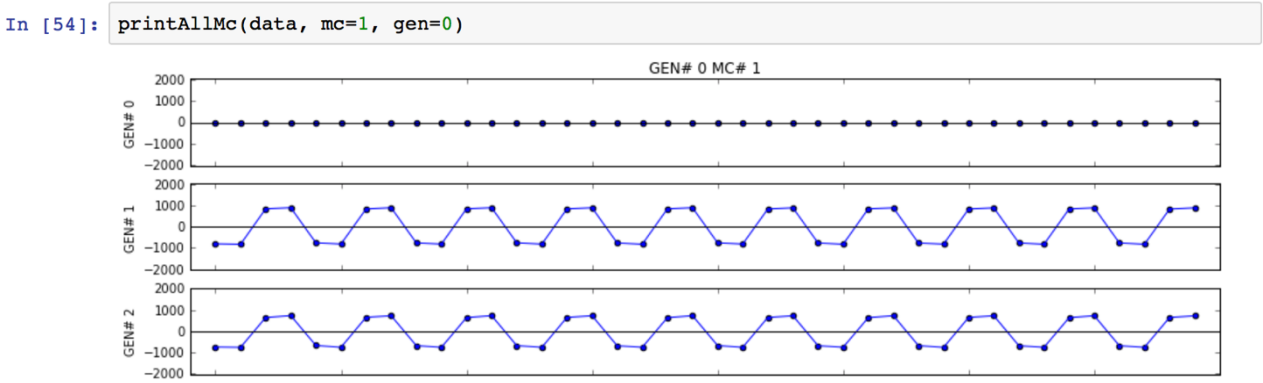

Рисунок 5 – визуализация исходных данных после реконструкции для проверки корректности исходных данных

Отличие такого подхода заключается в том, что за счет нелинейности исполняемого кода появляется возможность оперативно анализировать получаемые данные во время отладки без необходимости линейного внесения изменений в код отладочного программного обеспечения, которое затем потребовало бы повторения всего набора действий эксперимента, что позволяет существенно сэкономить время разработки.

4. Заключение

На примере разработки программного обеспечения для многопроцессорной архитектуры измерительной системы персонального электроимпедансного маммографа (ПЭМ) показано, что для обеспечения

синхронизации исполнения кода с точностью до 1 такта несущей частоты можно использовать состояния «сна» микроконтроллеров с последующим пробуждением с помощью синхронной обработки прерываний на канале синхронизации. Такой подход позволяет гарантировать предсказуемость времени исполнения в системах, где важна синхронизация во времени.

Также описан подход использования среды разработки Jupyter notebook на языке программирования Python, как пример возможного ускорения процесса разработки и отладки за счет комбинации возможностей асинхронного выполнения блоков кода и всей мощности и простоты языка на более высоком уровне абстракции, необходимом для анализа получаемых данных непосредственно во время проведения экспериментов.

Работы проведены при поддержке ФГБУ «Фонд содействия развитию малых форм предприятий в научно-технической сфере» (Фонд содействия инновациям).

Литература

1. O.V. Trokhanova, M.B. Okhapkin and A.V. Korjenevsky. Dual-frequency electrical impedance mammography for the diagnosis of non-malignant breast disease. *Physiol. Meas*., 2008, Vol. 29, pp. S331-S344
2. V. Cherepenin, A. Karpov, A. Korjenevsky, V. Kornienko, Y. Kultiasov, M. Ochapkin, O. Trochanova and D. Meister. Three-dimensional EIT imaging of breast tissues: system design and clinical testing. *IEEE Trans. Medical Imaging*, 2002, Vol. 21(6), pp. 662-667
3. Barber D. C., Brown B. H., Freeston I. L. Imaging spatial distribution of resistivity using applied potential tomography. *Electronics Letters*, 1983, Vol.19(22), pp.933-935
4. Brown B. H., Barber D. C., Segar A. D. Applied potential tomography: possible clinical applications. *Clin. Phys. Physiol. Meas*., 1985, Vol.6(2), pp.109-121

5. Создание профессиональных приложений для компьютеров с помощью IDE NetBeans [электронный ресурс]. Режим доступа: https://netbeans.org/features/java/index_ru.html
6. Пакет Anaconda [электронный ресурс]. Режим доступа: https://anaconda.org/anaconda/python

Ссылка на статью:
И. К. Лакеев, А. В. Корженевский, Т. С. Туйкин. Разработка программного обеспечения для многопроцессорной архитектуры персонального электроимпедансного маммографа ПЭМ. Журнал радиоэлектроники [электронный журнал]. 2017. №12. Режим доступа: http://jre.cplire.ru/jre/dec17/9/text.pdf