МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет компьютерных наук

Кафедра цифровых технологий

Цифровой широкополосный стетоскоп

ВКР Бакалаврская работа

02.03.01 Математика и компьютерные науки

Распределенные системы и искусственный интеллект

Допущено к защите в ГЭК __ . _____ . 2017 г.

Зав. кафедрой _____ *С.Д. Кургалин, д. ф.-м. н., профессор*

Обучающийся _____ *А.А. Родионов, 4 курс, д/о*

Руководитель _____ *Я.А. Туровский, к. мед. н, доцент*

Воронеж, 2017

Содержание

1 Введение

1.1 Постановка проблемы

Аускультация (выслушивание) звуков, исходящих от различных органов - одна из областей медицинской диагностики. Прибор для выслушивания звуков называется стетоскоп. Стетоскопы бывают акустические и электронные. Акустический стетоскоп передает звук от пациента непосредственно в ухо врачу. У электронных есть микрофон, который передает звук либо через динамики врачу, либо записывает для дальнейшего анализа.

Проблемой существующих на рынке электронных стетоскопов является невысокое качество звука. Под невысоким качеством звука подразумевается низкая частота дискретизации получаемого на выходе сигнала и как следствие неспособность выдавать данные о высокочастотном диапазоне звука (в частности ультразвука).

Эти стетоскопы теряют массу информации, которая может быть полезна врачам для осуществления медицинской диагностики пациентов.

1.2 Цель

Целью данной курсовой работы является создание доступного и простого в производстве цифрового стетоскопа. Стетоскоп должен быть способен регистрировать высокое качество звука. Он должен иметь высокую частоту дискретизации (600 кГц) и способен регистрировать сигнал в ультразвуковом диапозоне (до 100-300кГц)

В данной работе описывается создание ультразвукогого стетоскопа.

В ходе работы был создан рабочий прототип прибора. Прибор позволяет получать сигнал от ультразвукового микрофона. Также было написано программное обеспечение к этому прибору. Програмное обеспечение позволяет позволяет записывать и анализировать звуковые сигналы в реальном времени. Есть возможность рассмотреть различные характеристики сигнала, такие как спектр Фурье, скользящее среднее. Также есть возможность записывать аудиосигнал на жесткий диск для его последующей обработки.

1.3 Применение прибора на практике

Устройство планируется использоваться в медицине: анализ ультразвуковой составляющей звука от сердца и легких.

Данный прибор может использоваться в медицине для получения и анализа сигнала высокого качества. Например звук сердца и лёгких.

Это устройство поможет лучше анализировать звук сердца. С помошью визуализации сигнала можно получить больше информации о звуке внутренних органов, чем простое прослушивание.

Также прибор можно использовать в других областях, где необходим анализ ультразвука.

Например изучение дельфинов или летучих мышей.

Сердечно-сосудистые заболевания - самая распространенная причина смерти в мире по данным Всемирной Организации Здравоохранения (ВОЗ). Также рас-

пространенными являются заболевания легких. Своевременное наблюдение за состоянием сердца и легких, и обнаружение заболеваний - важная задача здравоохранения.

2 Описание аппаратной части разработанной системы

Разработанная информационная система состоит из следующих частей:

1. Врач
2. Пациент (диагностируемый)
3. Микрофон
4. Усилитель
5. Аналого цифровой преобразователь
6. Компьютер
7. Удаленный сервер для высокопроизводительных вычислений

Для подавления шумов и лучшей передачи звука от сердца, легких и других органов к микрофону присоединяются мембрана и соединительная трубка от аналогового стетоскопа. Сигнал с микрофона подается на усилитель. С усилителя сигнал подается на Аналого цифровой преобразователь (АЦП). Аналого цифровой преобразователь подключается к компьютеру через USB-порт и передает данные программе.

Краткая схема устройства:

Микрофон $\rightarrow$ Усилитель $\rightarrow$ АЦП $\rightarrow$ Компьютер

Рис. 1: Собранный прототип

2.1 Описание микрофона

В качестве микрофона был выбран SWEN MK-200.

Микрофон был вынут из стандартного корпуса, чтобы лучше соединиться с трубкой, ведущей к мембране.

Чувствительность, дБ	-60 ± 3
Диапазон частот, Гц	50 – 16 000
Размер микрофонного модуля, мм	9×7
Тип разъема	мини-джек Ø 3,5 мм (3 pin)
Длина кабеля, м	1,8
Вес, г	63

Таблица 1: Технические характеристики SWEN MK-200

2.2 Описание усилителя

Усилитель для микрофона был создан самостоятельно в рамках данной работы.

Была выбрана следующая схема усилителя:

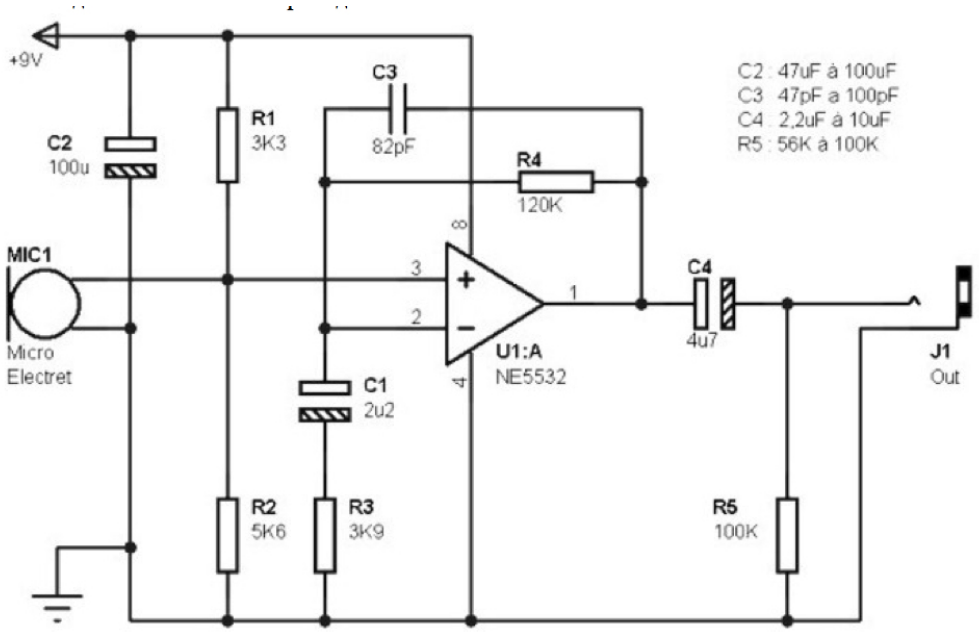


Рис. 2: Схема усилителя сигнала

В качестве операционного усилителя был выбран **MCP6022** от производителя Microchip. Это усилитель типа Rail-to-Rail SO-8.

SOIC или просто SO (small-outline-integrated-circuit), а также SOP (Small-Outline Package) корпус микросхем, предназначенный для поверхностного монтажа, занимающий на печатной плате на 30-50% меньше площади чем аналогичный корпус DIP, а также имеющий на 50-70% меньшую толщину. Обычно в обозначении также указывается число выводов.

В данный усилитель встроены High-Pass и Low-Pass фильтры. High-Pass фильтрует частоты сигнала меньше 1Гц. Low-Pass фильтрует частоты выше 100кГц. Меняя конденсатор C3, можно менять частоту среза LowPass фильтра. Усиление схемы зависит от резисторов R3 и R4. На текущий момент усиление составляет порядка 100.

Ниже приводятся параметры операционного усилителя.

Полоса частот	10МГц
Уровень шума	8.7 нВ/$\sqrt{}$Гц
Количество каналов	2
Напряжение питания	2.5В — 5.5В
Напряжение смещения	$\pm 500 \mu V$
Гармонические искажения	0.00053%
Температурный диапазон	-40°C — +85°C
Тип корпуса	SO-8

Таблица 2: Технические характеристики операционного усилителя MCP6022

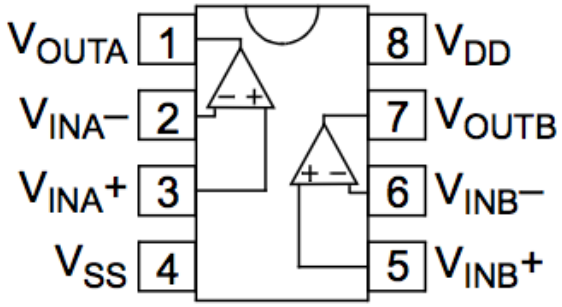
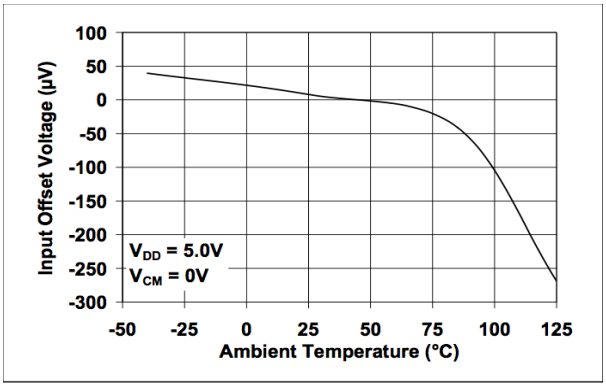
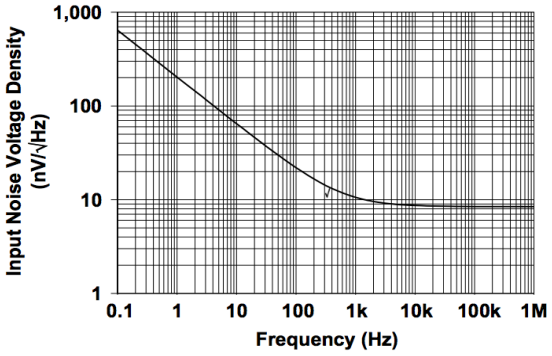


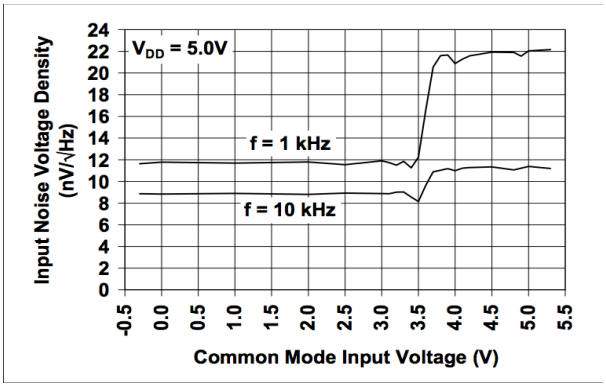
Рис. 3: Распиновка операционного усилителя



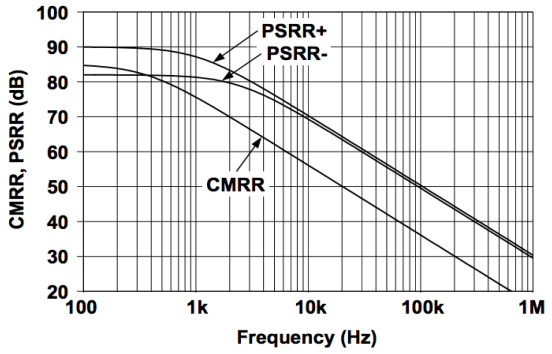
(a) Напряжение смещения — Температура



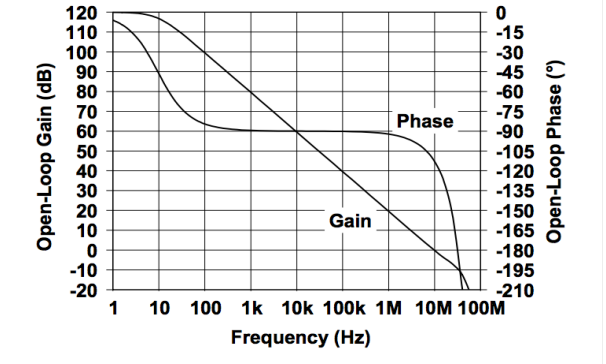
(b) Шум — Частота



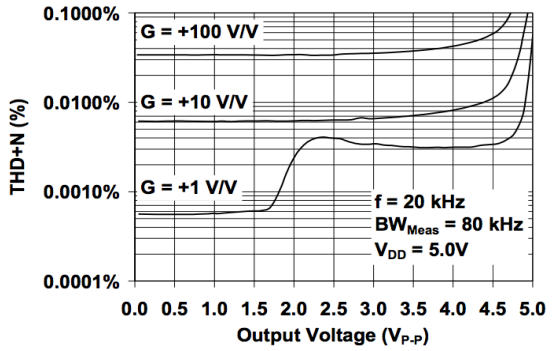
(c) Шум - Напряжение смещения



(d) CMRR — Частота



(e) Коэффициент усиления - Частота



(f) Гармонические искажения — Вых. напряжение (f=20кГц)

Рис. 4: Параметры операционного усилителя

Разработка платы усилителя велась в программе Sprint Layout. Ниже представ-

лен скриншот макета платы из этой программы.

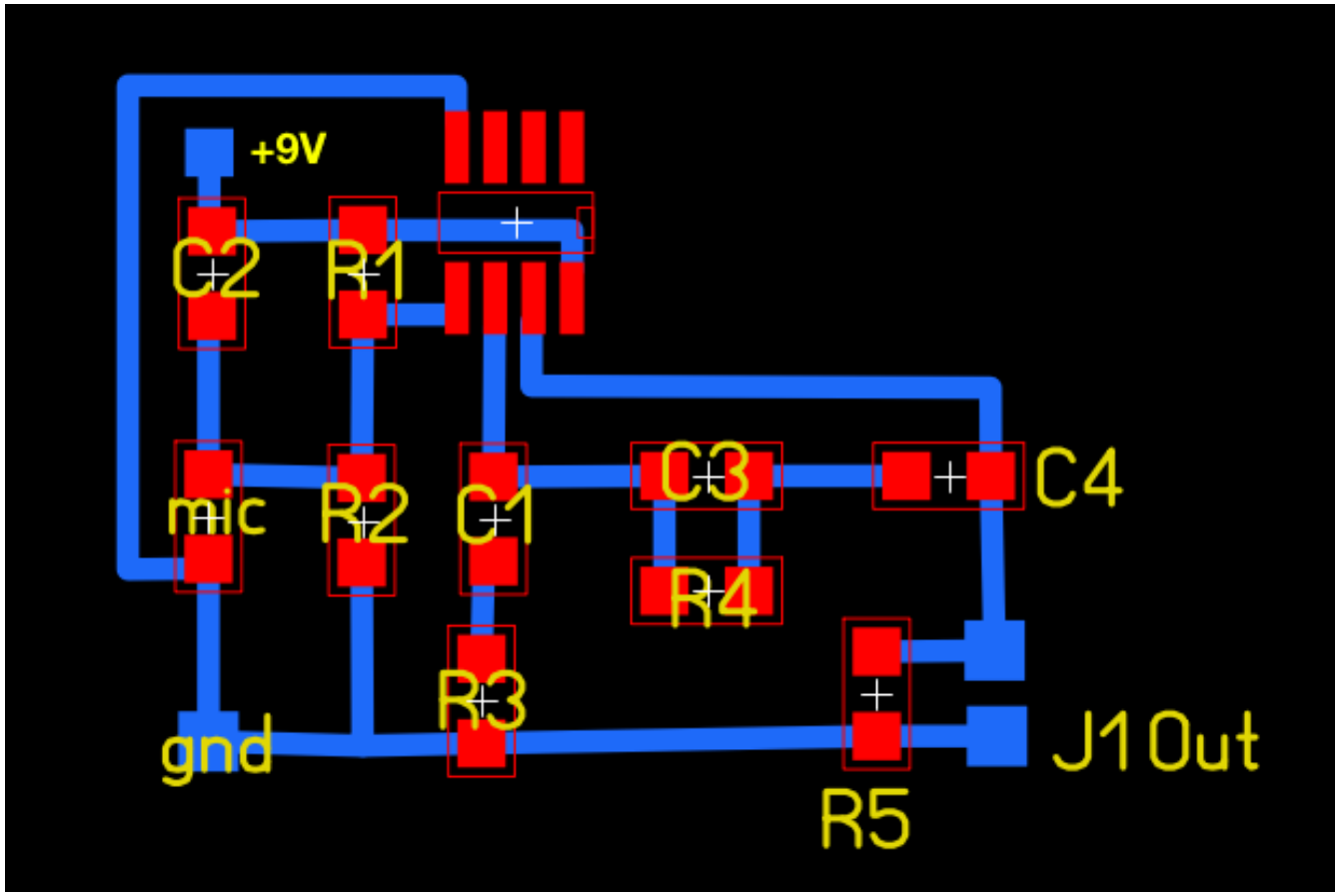


Рис. 5: Схема печатной платы усилителя

Печатная плата изготавливалась при помощи печати на лазерном принтере методом травления текстолита.

2.3 АЦП

Предназначение АЦП – преобразование непрерывных (аналоговых) входных сигналов в цифровую форму для дальнейшей обработки с помощью компьютера.

В качестве аналого цифрового преобразователя были опробованы 3 устройства:

1. LCard E14-140M
2. ЗАО "Руднев-Шиляев"ЛА-н10-12USB
3. Arduino Due

2.3.1 LCard E14-140M

Количество каналов	16 дифференциальных или 32 с "общей землей"
Максимальная частота дискретизации	200 кГц
Объем буффера памяти	64 Кбайт
Разрядность	14бит
Эффективная разрядность	13,3 бит (100 кГц, диап. изм 2,5 В.)
Входное сопротивление	Не менее 10 МОм
Диапазоны входного напряжения	$\pm 10V; \pm 2.5V; \pm 0.6V; \pm 0.15V$
Синхронизация	От внешнего синхросигнала, по уровню аналогового сигнала, внутренняя. Возможна многомодульная.
Защита входов	30 В (питание вкл.); 10 В (питание выкл. и в режиме suspend)

Таблица 3: Технические характеристики АЦП LCard E14-140M

При работе с данным АЦП возникли трудности при написании программного обеспечения. Средства разработки ПО для данного ацп давно не обновлялись. Работа с АЦП плохо документирована производителем и приведено мало примеров использования. Тем не менее, удалось запустить данный АЦП на одном

из примеров производителя. Данный пример позволяет записывать данные с АЦП в бинарный файл.

2.3.2 ЛА-н10-12USB

Число аналоговых входов	2
Минимальная частота дискретизации	1.25МГц
Максимальная частота дискретизации	80МГц
Объем буффера памяти	$2 \times 10^{19} = 524288$
Разрядность	12бит (4096 значений)
Входное сопротивление	50Ом
Разъем	BNC
Диапазоны входного напряжения	$\pm 2V; \pm 1V; \pm 0.4V; \pm 0.2V$
Защита по входному напряжению	$\pm 5V$
Дифференциальная нелинейность	± 1.2 МЗР
Интегральная нелинейность	± 1.5 МЗР
Ошибка сдвига	$\pm 0.15\%$
Интерфейс	USB
Потребляемая мощность	12В, 0.7А
Масса	400г

Таблица 4: Технические характеристики АЦП ЛА-н10-12USB

Интегральная нелинейность - отклонение по вертикальной оси точек реальной характеристики от идеальной характеристики преобразования, делящих пополам расстояние по оси абцисс между средними значениями пороговых уровней характеристики преобразования. Измеряется в процентах или МЗР.

Дифференциальная нелинейность - отклонение разности двух аналоговых сигналов от значения, соответствующего единице МЗР.

Для данного АЦП было написано програмное обеспечение, позволяющее принимать данные с АЦП и визуализировать их в реальном времени.

Проблемы данного АЦП:

Данный АЦП работает в режиме старт-стоп. Другими словами он периодически производит запуск, сбор данных(в буффер) и остановку. Это один полный цикл сбора данных. При этом полезное время - это сбор данных, а старт и стоп - бесполезное.

Время, за которое АЦП совершает полный цикл сбора данных и соотношение полезного и бесполезного времени отличается в зависимости от частоты дискретизации и размера буффера. Были произведены замеры времени для различных значений частоты дискретизации и размера буффера и были составлены следующие таблицы.

Во всех таблицах ось X: размер буффера, ось Y: частота дискр.

	2^19	2^18	2^17	2^16	2^15	2^14	2^13	2^12	2^11	2^10
8*10^7	190	96	48	25	13	7	4	2	2	1
8*10^6	348	174	88	44	23	12	6	4	2	2
8*10^5	851	426	214	107	54	27	14	8	4	3

Рис. 6: Время на полный цикл (мс)

	2^19	2^18	2^17	2^16	2^15	2^14	2^13	2^12	2^11	2^10
8*10^7	6.55	3.27	1.6	0.81	0.4	0.2	0.1	0.05	0.02	0.01
8*10^6	65.5	32.7	16.3	8.19	4.09	2.04	1.02	0.51	0.25	0.12
8*10^5	655	327	163	81.9	40.9	20.4	10.2	5.12	2.56	1.28

Рис. 7: Полезное время (мс)

	2^19	2^18	2^17	2^16	2^15	2^14	2^13	2^12	2^11	2^10
8*10^7	0.034	0.034	0.033	0.032	0.031	0.029	0.025	0.025	0.010	0.010
8*10^6	0.188	0.188	0.185	0.186	0.178	0.170	0.170	0.128	0.125	0.060
8*10^5	0.770	0.768	0.762	0.765	0.757	0.756	0.729	0.640	0.640	0.427

Рис. 8: Отношение полезного времени к полному

Как видно из таблицы, даже при самых оптимальных значениях размера буфера и частоты дискретизации, отношение полезного времени к полному составляет 77%. (полезное время 655мс, полное - 851мс) Тоесть в конце каждого цикла образуется дырка длиной 196мс. При таком долгом времени на перезапуск теряется очень много информации, что неприемлимо для данного проекта. Поэтому было принято решение заменить его. Данный же подходит для коротких сигналов (меньше секунды), которые нужно оцифровывать в сверхвысоком качестве.

2.3.3 Arduino Due

Число аналоговых входов	12
Максимальная частота дискретизации	1МГц
Объем буффера памяти	512 KB
Разрядность	12бит (4096 значений)
Рабочее напряжение	3.3V
Диапазоны входного напряжения	7-12V
Защита по входному напряжению	6-16V
Интерфейс	USB
Микроконтроллер	AT91SAM3X8E
Масса	36г

Таблица 5: Технические характеристики АЦП Arduino Due

Самым оптимальным вариантом ацп для данного проекта оказался Arduino Due. Он сочетает в себе как простоту в использовании так и возможность оцифровывать сигнал высокого качества. Максимальная частота дискретизации Arduino Due составляет 1МГЦ. В ходе данной работы удалось достичь максимума в 672кГц. Обычно среднее значение частоты дискретизации составляло 666кГц. Максимальное значение частоты дискретизации зависит также от производительности компьютера. Данное устройство тестировалось на MacBook Air 2014.

3 Описание программного обеспечения

В рамках данной работы было написано програмное обеспечение для работы с ультразвуковым стетоскопом. Были разработаны программы под 2 разных АЦП: ЗАО "Руднев-Шиляев"ЛА-н10-12USB и Arduino Due. Также была написана программа для вычисления быстрого преобразования фурье на удаленном высокопроизводительном сервере с помощью технологии NVidia CUDA.

Ниже приводятся описания программ для работы с двумя разными АЦП а также описание программы, запускаемой на CUDA-сервере.

3.1 Програмное обеспечение для ЛА-н10-12USB

Програмное обеспечение написано на языке C# для платформы Windows.

В функционал програмного обеспечения входит:

- Получение сигнала от АЦП ЛА-н10-12USB
- Отображение сигнала в реальном времени на экране в виде графика
- Отображение спектра сигнала полученного в результате быстрого преобразования Фурье
- Отображение скользящего среднего сигнала
- Возможность записи сигнала на жесткий диск для последующей обработки

3.1.1 Инструкция по установке ПО

- Скачать драйверы для АЦП ЛА-н10-12USB по ссылке: `http://rudshel.ru/software.html`
- Запустить и установить файл `RShUSBDriver_XP_Vista_7_8_x86_x64_SDK2_2.0.x.y.exe` (драйверы для USB устройств)
- Запустить и установить файл `RshSDKSetup_2.0.x.y.exe` (средства разработки ПО (SDK))
- Скачать проект для работы со стетоскопом по ссылке `https://github.com/tandav/ultrasonic-stethoscope/tree/master/Rudshel/forms-timer-label`
- Скомпилировать проект с помощью IDE Microsoft Visual Studio.

3.1.2 Описание интерфейса пользователя ПО

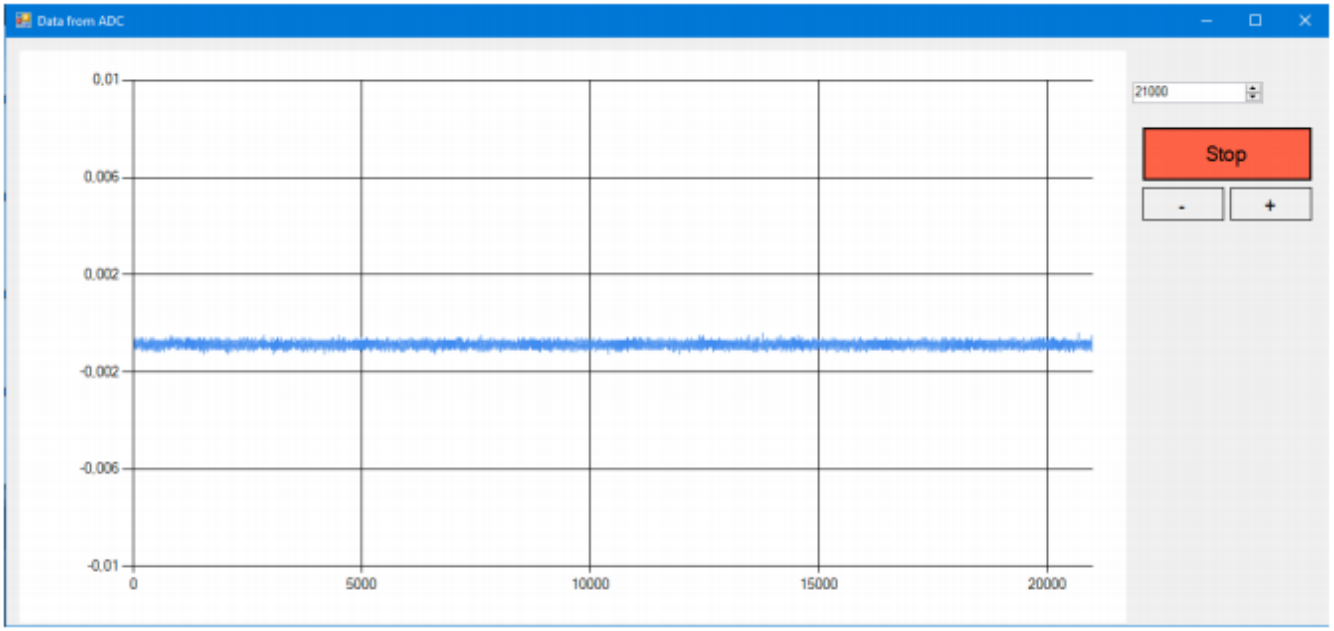


Рис. 9: Скриншот работы программы

В центре находится график сигнала с АЦП. Справа находится кнопка старт/стоп, которая управлет запуском и остановкой сбора сигнала с АЦП. Справа вверху есть числовое поле, которое отвечает за количество точек по горизонтальной оси на графике. Кнопки "+/-"Отвечают за увеличение и уменьшение размеров графика (zoom).

3.1.3 Описание кода программы

В начале программы задаются параметры размера буффера и частоты дискретизации (RATE).

```
const uint   BSIZE              = 524288; // размер буффера
const double RATE               = 8.0e+7;
```

Далее при нажатии пользователем на кнопку старт, программа начинает инициализацию АЦП. Она проверяет подключено ли устройство, проверяет его работоспособность и начинает сбор данных.

За инициализацию отвечают следующие строчки в коде:

```
//========================== ИНИЦИАЛИЗАЦИЯ ================
st = device.EstablishDriverConnection(BOARD_NAME); //загрузка и
подключение к библиотеке абстракции устройства
if (st != RSH_API.SUCCESS) SayGoodBye(st);
st  = device.Connect(1); //Подключаемся к устройству. Нумерация с 1.
if (st != RSH_API.SUCCESS) SayGoodBye(st);
p.startType           = (uint)RshInitMemory.StartTypeBit.Program;
//Запуск сбора данных программный.
```

```
p.bufferSize          = BSIZE; //Размер внутреннего блока данных,
по готовности которого произойдёт прерывание.
p.frequency           = RATE;  //Частота дискретизации.
p.channels[0].control = (uint)RshChannel.ControlBit.Used;  //Сделаем
0-ой канал активным.
p.channels[0].gain    = 10; // коэффициент усиления для 0-го канала.
[1, 2, 5, 10] ~ [+-0.2V, +- 0.4V, +-1V, +- 2V]
st = device.Init(p); //Инициализация устройства (передача
выбранных параметров сбора данных)
if (st != RSH_API.SUCCESS) SayGoodBye(st); //После инициализации
неправильные значения в структуре будут откорректированы.
```

Далее программа ждет пока пользователь нажмет кнопку старт. При нажатии на кнопку старт вызывается функция `button1_Click`, которая в свою очередь запускает новый процесс `backgroundWorker1_DoWork`. Этот процесс собирает данные с АЦП в бесконечном цикле пока пользователь не нажмет на кнопку стоп. Он записывает значения с АЦП в структуру данных очередь, доступ к которой имеет UI-процесс (процесс, отвечающий за отображение и обработку элементов пользовательского интерфейса). По мере обновления данных UI процесс обновляет график.

За старт, сбор данных и остановку АЦП отвечает следующий блок кода:

```
double[] buffer = new double[p.bufferSize]; //Получаемый из платы буфер.
uint waitTime = 100000; // Время ожидания(в миллисекундах) до
наступления прерывания. Прерывание произойдет при полном заполнении
буфера.  // default = 100000
while (getting_data)
```

```
{
    stopwatch.Restart();

    st = device.Start(); // Запускаем плату на сбор буфера.
    if (st != RSH_API.SUCCESS) SayGoodBye(st);

    st = device.Get(RSH_GET.WAIT_BUFFER_READY_EVENT, ref waitTime);
    if (st != RSH_API.SUCCESS) SayGoodBye(st);

    st = device.GetData(buffer); // very big amount of data
    if (st != RSH_API.SUCCESS) SayGoodBye(st);

    device.Stop();

    // Queue Dequeue and Enqueue implementation with arrays
    double[] values_to_draw_copy = (double[])values_to_draw.Clone();
    for (int i = 0; i < x_axis_points - r_buffer_size; i++)
        values_to_draw[i] = values_to_draw_copy[r_buffer_size + i];

    for (int i = 0; i < r_buffer_size; i++)
        values_to_draw[x_axis_points - r_buffer_size + i] =
        buffer[BSIZE / r_buffer_size * i];

    stopwatch.Stop();
    Console.WriteLine("GetData() time:\t" +
    stopwatch.ElapsedMilliseconds + "ms");
}
```

Также процесс backgroundWorker замеряет время каждого цикла сбора данных. (stopwatch.Restart() и stopwatch.Stop()) Это время процесс пользовательского интерфейса выводит на экран.

3.1.4 Измерение времени работы АЦП

Как уже было сказано выше, в зависимости от различных значений размера буффера и частоты дискретизации у АЦП уходит различное время на перезапуск и сбор данных. Эти вычисления производились при помощи стандартной функции-секундомера из стандартной библиотеки C#.

3.1.5 Паралельные вычисления в программе

В программе используются паралельные вычисления. Присутсвуют два потока. Один поток работает с АЦП: принимает данные и записывает во временный буфер. Другой поток - отрисовывает данные из буффера на графике а также обрабатывает команды пользователя.

Паралельные вычисления используются из-за необходимости непрерывно получать данные с АЦП, чтобы избежать потери данных. Также, если не использовать паралельные вычисления, то приложение может не реагировать на команды пользователя из за того что процессор будет занят работой с АЦП.

3.2 Програмное обеспечение для Arduino Due

Програмное обеспечение для микроконтроллера Arduino Due состоит из программы, запускаемой на Arduino и программы, запускаемой на компьютере.

Возможности программного обеспечения:

1. Прием сигнала от АЦП Arduino через Native-USB порт
2. Визуализация сигнала в реальном времени
3. Запись на диск в бинарный файл
4. Отправка записаного сигнала на CUDA-сервер для вычисления быстрого преобразования фурье.
5. Возможность посчитать быстрое преобразование фурье локально

3.2.1 Программа для запуска на Arduino

Для того, чтобы ардуино работало в нужном режиме, на него нужно записать следующую программу:

```
#undef HID_ENABLED
volatile int bufn,obufn;
uint16_t buf[4][256];   // 4 buffers of 256 readings

void ADC_Handler(){     // move DMA pointers to next buffer
  int f=ADC->ADC_ISR;
  if (f&(1<<27)){
   bufn=(bufn+1)&3;
   ADC->ADC_RNPR=(uint32_t)buf[bufn];
   ADC->ADC_RNCR=256;
  }
```

```
}

void setup(){
  SerialUSB.begin(0);
  while(!SerialUSB);
  pmc_enable_periph_clk(ID_ADC);
  adc_init(ADC, SystemCoreClock, ADC_FREQ_MAX, ADC_STARTUP_FAST);
  ADC->ADC_MR |=0x80; // free running

  ADC->ADC_CHER=0x80;

  NVIC_EnableIRQ(ADC_IRQn);
  ADC->ADC_IDR=~(1<<27);
  ADC->ADC_IER=1<<27;
  ADC->ADC_RPR=(uint32_t)buf[0];   // DMA buffer
  ADC->ADC_RCR=256;
  ADC->ADC_RNPR=(uint32_t)buf[1]; // next DMA buffer
  ADC->ADC_RNCR=256;
  bufn=obufn=1;
  ADC->ADC_PTCR=1;
  ADC->ADC_CR=2;
}

void loop(){
  while(obufn==bufn); // wait for buffer to be full
  SerialUSB.write((uint8_t *)buf[obufn],512); // send it - 512 bytes
  = 256 uint16_t
```

```
  obufn=(obufn+1)&3;
}
```

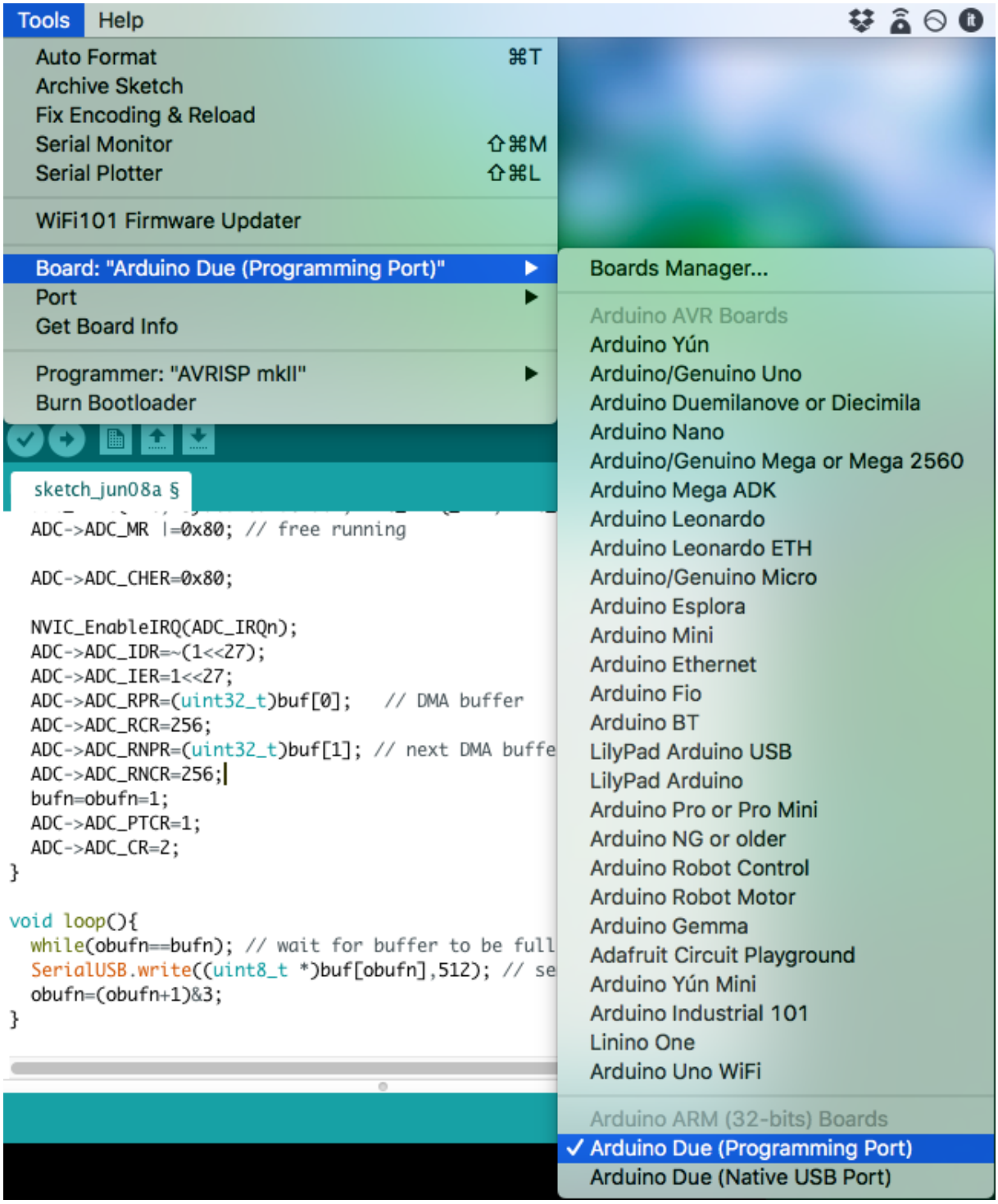
Данный код запускает АЦП в так называемом Free Running Mode. Это самый быстрый режим работы данного АЦП. Его особенность заключается в том, что нет необходимости вручную собирать данные через определенное количество времени. Программа пользователя запускает только первый сбор данных, а дальше АЦП сам автоматически начинает преобразование сразу как только закончилось предыдущее. Таким образом не возникает простоев АЦП и не теряются данные.

Также здесь используется прямой доступ к памяти микроконтроллера AT91SAM3X8E. (Direct Memory Access / DMA). Этот режим позволяет обращаться напрямую к памяти микроконтроллера, таким образом не отнимая процессорное время самого микроконтроллера. Процессор не прерывается на обработку запроса к памяти и постоянно занят сбором данных с АЦП. Это позволяет приблизится к максимальной частоте дискретизации.

Чтобы установить данное ПО на Arduino Due, необходимо скачать Arduino IDE с официального сайта Arduino: `https://www.arduino.cc/en/Main/Software`

Устройсво нужно подключить к компьютеру через USB порт. Данное устройство имеет 2 USB порта: программируемый и native. Чтобы записать программу нужно подключаться через програмируемый.

В Arduino IDE в меню Tools - Board нужно выбрать "Arduino Due (Programming Port)"как показано на скриншоте ниже.



В главное окно программы нужно вставить код, приведенный выше и нажать кнопку Upload. После этого программа будет успешно записана на микроконтроллер.

3.2.2 Программа для запуска на компьютере

Данное ПО написано на языке Python и может быть запущено на любой операционной системе. Используются библиотеки PyQtgraph, PyQt, numpy, matplotlob и pyserial. Для запуска программы необходимо установить эти библиотеки:

1. Скачать и установить последнюю версию python с официального сайта: `https://www.python.org/downloads/`
2. открыть командную строку и установить необходимые библиотеки с помощью комманд:
3. sudo pip install pyqtgraph
4. pip install pyqt5
5. pip install numpy
6. pip install matplotlib
7. pip install pyserial
8. скачать саму программу для работы с ацп по ссылке: `https://github.com/tandav/ultrasonic-stethoscope/blob/master/arduino/app.py`

Даллее в коммандной строке необходимо перейти в директорию, где находится скачанная программа `app.py`. Теперь нужно подключить устройство к компьютеру через Native-USB порт и запустить программу с помощью команды:

```
python app.py
```

При запуске программы можно указывать параметры даунсемплинга (downsampling) Даунсемплинг позволяет отрисовывать на графике не все значения с ацп, а в

N раз меньше. Это позволяет увеличить скорость отрисовки (Более высокое значение FPS (frames per second, количество кадров в секунду), но при этом теряется точность сигнала, можно не увидеть очень высоие частоты на графике. Значение даунсемплинга задается с помощью аргумента `-d`. Допустим, команда

```
python app.py -d 100
```

запустит программу со значением параметра downsampling=100.

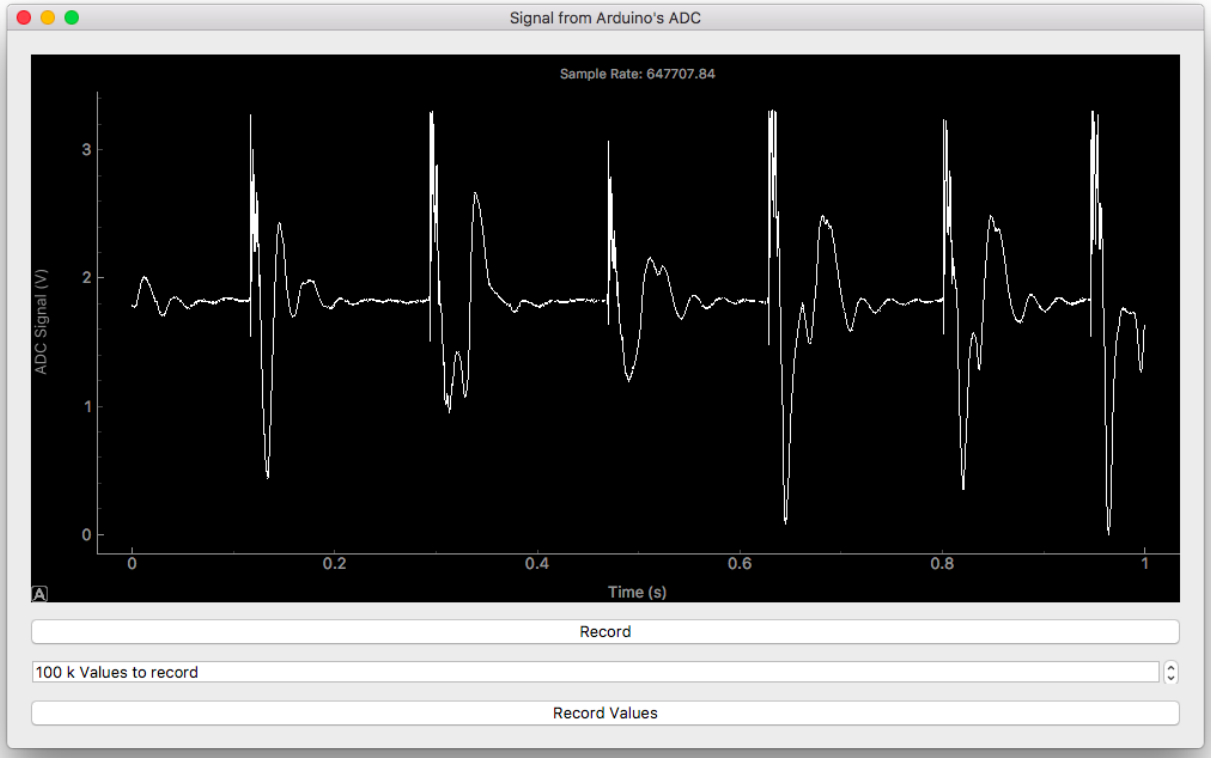


Рис. 10: Скриншот работы программы

Для записи данных в файл и отправки их на сервер нужно выбрать сколько значений записывать и нажать кнопку Record Values.

Основные компоненты программы:

- Класс `SerialReader` - принимает данные с ацп через serial порт. В этом классе есть 2 основных метода: метод `run()`, который в бесконечном цикле собирает данные с АЦП и, если нажата кнопка записи, записывает данные в файл. Метод `get(num, downsample=1` - возвращает num последних значений с ацп. Этот метод использует второй класс `adc_chart`.
- Класс `adc_chart` - работает с интерфейсом пользователя. Основные методы:
  - `init_ui` - инициализация пользовательского интерфейса
  - `updateplot` - метод получает последние значения с АЦП и отрисовывает их на графике
  - `on_record_values_button` - обработка кнопки записи сигнала.
  - `closeEvent` - обработка закрытия окна программы. Завершает все запущенные процессы.
- `send_to_cuda` - метод вызываемый, когда завершена запись данных. Отправляет данные на CUDA-сервер.
- `calc_fft_localy` - метод позволяющий посчитать быстрое преобразование фурье на локальном компьютере, без отправки на сервер. Для вычисления и отрисовки используются библиотеки Numpy и Matplotlib.

3.3 Програмное обеспечение для CUDA-сервера

Програмное обеспечение для сервера написано на языке C# под операционную систему Windows. ПО позволяет считать быстрое преобразование фурье

на видеокартах NVidia с помощью технологии CUDA (библиотека для работы с высокопроизводительными математическими вычислениями на видеокартах NVidia).

Возможности данного ПО:

1. Прием сжатого сигнала от клиента
2. Декомпрессия сигнала и вычисление быстрого преобразования фурье (FFT)
3. Построение графиков сигнала и его спектра и сохранение в .png файл.
4. Отправка полученных изображений обратно клиенту

Для запуска, необходимо поместить в папку с программой `.dll` - библиотеку `CUFT.dll` для работы с видеокартой. Библиотеку можно скачать по ссылке `https://github.com/tandav/ultrasonic-stethoscope/blob/master/Server/CUFT-lib/CUFT.dll`

Основные методы главного класса программы:

1. `receive_and_write_to_file` - принимает сжатый сигнал от клиента и записывает его в файл.
2. `Decompress` - распаковка .gz архива с сигналом. Метод возвращает массив с сигналом.
3. `Main` - основной метод программы, в котором считается fft и вызываются остальные методы.
4. `save_pngs` - Построение графиков сигнала и его спектра и сохранение в .png файл(ы).

В методе Main программа в бесконечном цикле пытается принять данные от клиента. Если клиент подключен, то сигнал принимается. Если сигнал большой, то он делится на блоки. Для каждого блока считается FFT и создается график спектра сигнала, который записывается в отдельный файл-изображение. Когда все блоки обработаны, сервер продолжает ждать приема нового сигнала.

4 Заключение

В результате данной работы был создан рабочий прототип устройства, позволяющего анализировать в реальном времени звуковой сигнал высокого качества. Сигнал записывается с частотой дискретизации 660кГц. Данная частота позволяет анализировать ультразвуковой сигнал до 330кГц.

Данное устройство предназначено для получения дополнительной информации из звука поступающего от сердца, легких и других внутренних органов, которую нельзя услышать на обычном стетоскопе. Данная дополнительная информация может быть использована врачом для осуществления более качественной медицинской диагностики.

Сфера применения не ограничена медициной, устройство позволяет анализировать любые типы ультразвуковых сигналов.

Развитие проекта можно продолжить в направлении улучшения качества звука. Для этого нужно использовать более дорогие микрофон и усилитель. Также нужно произвести опрос врачей о том, какие именно характеристики звука со стетоскопа важны.

Улучшений в програмном обеспечении можно достигнуть путем оптимизации алгоритмов распараллеливания на нескольких ядрах процессора или на видеокарте. Также, на основе информации от докторов, можно сделать систему распознавания различных забовалеваний легких и сердца.

Разработка данного проекта велась с помощью системы контроля версий git. Исходный код програмного обеспечения, этапы создания и документация доступны по адресу: `https://github.com/tandav/ultrasonic-stethoscope`

Ссылки на источники

[1] Аналого цифровой преобразователь ЛА-н10-12USB

http://www.rudshel.ru/show.php?dev=14

[2] Драйверы и програмное обеспечение для устройств ЗАО "Руднев-Шиляев"

http://rudshel.ru/software.html

[3] Документация по программированию устройств ЗАО "Руднев-Шиляев"

http://www.rudshel.ru/soft/SDK2/Doc/CPP_USER_RU/html/index.html

[4] Руководство пользователя ЛА-н10-12USB

http://www.rudshel.ru/pdf/LA-n10-12USB(y).rar

[5] Внешний USB АЦП/ЦАП E14-140-M

http://www.lcard.ru/products/external/e-140m)

[6] Схема усилителя для микрофона

http://full-chip.net/analogovaya-elektronika/70-usilitel-dlya-elektretnogo-mikrofona-s-nizkim-urovnem-shuma-shema.html

[7] Руководство пользователя и технические характеристики операционного усилителя MCP6022

https://lib.chipdip.ru/291/DOC000291231.pdf

[8] Исходный код, документация и этапы создания проекта ультразвукового стетоскопа

https://github.com/tandav/ultrasonic-stethoscope