RemediUltraSound

Medical products for low-resource settings

0.0.0-cmake

generated on Thu Mar 12 2015 00:04:25

Contents

Chapter 1

Main Page

This project tries to create an open-source Ultasound platform. For more info see http://www.remedi.co. See licensing for the information about open source.

A high-level description of the system is given in the system_description. Some requirements have been created, but these are still very much open to discussion.

The compiled documentation can be read at http://remediosm.github.io/UltraSound/

Compiling

```
mkdir bld
cd bld
cmake ..
make
```

Generating documentation

```
cd bld
make doc
```

Chapter 2

Licensing

This page describes the licencing for the Remedi UltraSound system.

2.1 Licensing of documentation

Todo determine licensing

2.2 Licensing of source code

Todo determine licensing

Chapter 3

Requirements

This page lists all requirements for the Remedi UltraSound system.

This page lists all requirements for the Remedi UltraSound system. The requirements are based on the Open Source scanner `here`.

Requirements listed as shall (Remedi UltraSound shall) will indicate items that are required in the final product. Requirements listed as should (Remedi UltraSound should) will indicate items that, while not required, are especially important and justification must be made if they are omitted from the final product. Requirements listed as may (Remedi UltraSound may) will indicate optional items in the final product.

3.1 Remedi UltraSound shall

1. be a graphical user interface,
2. display live ultrasound images in a window that
   (a) shall be resizable,
   (b) shall allow scrolling (i.e. moving the image side-to-side and up/down in the window),
   (c) should be undockable (i.e. the window can be moved to a new location on the screen),
   (d) may be cloned so that multiple image windows could exist (overlapping and such) - If this is done, at least one shall be able to display live image data when the probe is scanning,
3. allow zooming of the image in the window,
4. allow restoring a zoomed image back to an unzoomed state,
5. allow saving of individual image data (backscatter data) to a file,
6. allow saving of a complete cine to a file,
7. allow arbitrary cine sizes (based on available memory),
8. have image processing controls that
   (a) shall allow for image contrast/intensity changes,
   (b) shall allow for image filtering changes
   (c) may allow for other gray-scale image processing,
   (d) may allow for pseudo-color image processing,
   (e) shall allow for gain correction with at least two control points (i.e. a line),
   (f) shall be adjustable via the GUI
9. allow entry of basic patient data (name, birthday, sex, date, physician/nurse name, etc.)

10. allow for control of probe parameters during live imaging that shall include
    (a) probe scan rate (frames per second),
    (b) probe sample frequency (depth mode),
    (c) probe pulse frequency,
    (d) probe pulser power,
    (e) probe unidirectional/bidirectional imaging control,
11. allow for controlling features of the scan converter that shall include
    (a) turning interpolation on/off,
    (b) turning image averaging on/off,
12. allow for multiple probes to be attached and a selection made, on the fly, for live imaging,
13. use the probe button (if available) to start/stop live imaging
14. allow printing of an image window (with all its contents) on a standard system printer and
    (a) shall print any available patient data on this printout,
    (b) shall print the current time/date,
15. allow image window contents to be copied into the system clipboard for pasting into other applications that may allow pasting of bitmaps.
16. have measurement capabilities that
    (a) shall allow measuring distance on an image,
    (b) shall allow at least two measurements to be taken,
    (c) may allow calculations to be done with the measurements (addition, subtraction, distance between, etc.),
    (d) may allow other measurement types:
        i. area of a circle or rectangle,
        ii. perimeter

3.2 Remedi UltraSound should

1. have hot-keys for all of the commonly used functions (to obviate the need for using a mouse once a live scan has begun),
2. save a PNG or other image type of the currently displayed image (with all processing and annotations) when a backscatter file is saved,
3. use modern, docking, window controls,
4. allow annotations to the image - these annotations shall include drawing arbitrary text, rectangles, circles and freehand lines and they should allow editing of those annotations once placed
5. allow for a complete database system to be used for storing patient data and records
6. support both A-mode and B-mode imaging,
7. allow for custom settings to be created and managed which includes the saving of
    (a) the current image processing controls and settings,
    (b) other data that may be entered at start up (such as the examiner's name),
8. incorporate a help system that may be context sensitive.

3.3 Remedi UltraSound may

1. allow the user to define what the probe button function will do (defaults to start/stop).

Chapter 4

System Description

Describes an UltraSound system from a High Level

Ultrasound imaging provides medical personnel the chance to non-instrusive diagnostics. The biggest advantage over other techniques is that it doesn't rely on electromagnetic radiation. It is used for a wide range of diagnostics. The average system in use in hospitals today can cost up to 40.000 euros. Although a basic version could be realized at a fraction of that cost using of-the-shelf hardware. Decomposition of an ultrasound system

According to sprab18a the following subsystems make up an ultrasound system

- Transducer
- Front-end processing
- Mid-end processing
- Back-end processing
- System controller
- Display
- Power supply

4.1 Transducer

The sensor of the system, usually consisting of multiple piezo-electric elements which can send ultrasonic pulses of 5-10 MHz.

4.2 Front-end processing

The front-end processing controls the analogue part of the system. It is responsible for forming the beams and converting the received beams to scan lines. All of this is controlled by the front-end controller.

4.3 Mid-end processing

The mid-end processing part of the system converts the raw scan lines into images. Ultrasound systems have multiple modes of operating. B-mode is the well known grayscale image. Color flow is super-imposed on the B-mode image and depicts the flow of blood. Doppler mode shows the flow of blood at a specific location.

4.4 Back-end processing

Back-end processing takes the output of the mid-end processing and tries using image processing algorithms to improve the quality by removing noise. It also converts the scanlines to raster data ready for display.

4.5 System controller

The system controller coordinates all subsystems and handles all interactions with the user.

Chapter 5

Workflow

This page describes the workflow for the Remedi UltraSound project.

5.1 Workflow

This project works following the git-flow branching model. Each feature is developed on a feature branch, branched of of develop. Check out http://nvie.com/posts/a-successful-git-branching-model/ for more info.

The below workflow is based on http://qq.is/tutorial/2011/10/23/git-flow-on-github.↩
html

5.2 Setting up

First clone the repository

```
git clone https://github.com/RemediOSM/UltraSound.git
```

Go into the repo

```
cd UltraSound
```

Setup the origin

```
git remote add upstream git@github.com:RemediOSM@UltraSound
```

Setup git flow (first install git flow if you haven't got it)

```
git flow init
```

And accept all the defaults

5.3 Starting on your feature

Create a new branch for your awesome feature

```
git flow feature start <my_great_feature>
```

Push the branch remote.

```
git flow feature publish <my_great_feature>
```

Commit your changes reguraly locally with descriptive messages.
Also push the changes back up to GitHub.

```
git push origin HEAD
```

5.4 Finish work

Create a pull request in the GitHub interface. In the pull request add usefull info. Click the send pull request to confirm you think you're done.

When your awesome feature is reviewed, sometimes additional changes are needed. Make them locally, commit and push them up to your branch.
Make sure your on your feature branch:

```
git checkout feature/<my_awesome_feature>
```

Do your development, commit and push the changes again. (see Starting on your feature).

5.5 Cleanup

When all your changes are agreed upon and merged by the project, your feature branch will be deleted. Locally you can finish your feature as well.

```
git flow feature finish
```

Chapter 6

Todo List

Page Licensing

determine licensing

determine licensing