Live Demo Setup Instructions:

Download the compiled Ulterius demo package on wiki.

Setup the SonixTouch machine:

1. Set IP address to 192.168.112.4 and subnet mask to 255.255.255.0
2. On B-Geom: Set all line densities to 128 and accumulator to 1.
3. On B-TX: Set pulse index A to 3 (turns of transducer).
4. Connect laptop or desktop by Ethernet to US machine.
5. Configure laptop's IP to 192.168.112.1 and subnet mask to 255.255.255.0
6. Run the Ulterius Demo on the laptop.
7. Once the demo is started press the connect buttom on the upper left corner of the interface.
8. Press the color button on the Ulterius device followed by B-mode image button.
9. Press the RF-beamformed option on the Ulterius interface.
10. Real time images will now be displayed.
11. To save data, enter a file name into the data_filename input on the interface. Press the save data button to record one image frame of rf data.
12. If image averaging is desired, press the enable averaging button and input number of frames to be averaged in below input tab.

PA Simulation Demo Instructions:

1. Download the PA Simulation Code package.
2. Run the generate_US_Simulation.m file to run Kai's simulation + my C++ rebeamformer. (Note, majority of code here is from the MUSiiC lab, I used the code mainly to assess my C++ rebeamformer).
3. Results between the two rebeamformers will be shown at the end.

PA Real Data Analysis Instructions:

1. Download the PA Real Data Analysis Code package.
2. Run the Analyze_Data.m file to run the optimized C++ rebeamformer on real PA RF data acquired from the Ulterius demo.
3. Results are displayed at the end.

Overview of PA Rebeamformer Algorithm and modified Ulterius (packaged in sdk6 on wiki):

Function call: double* pa_rebeamforming(const int ns, const int nl, double channelSpacing, double speedSound, double freq, double* rf)

Function input description:

1. rf = array of RF data (in double or int_16 format)
2. ns,nl = dimensions of RF data(# samples per line, # lines)
3. channelSpacing = distance between lines
4. speedSound = speed of sound (for depth calculations)
5. freq – freq of sampling.

This algorithm calculates a new time delay and then applies another round of dynamic beamforming on data. The beamforming effectively sums together elements in the array based on this time delay (where depth is roughly 1/2 that of the original beam-forming).

Output = rebeamformed rf array (of same dimension as input array)

Note: to speed up the algorithm, we computed the delay tables separately in our real RF data version. The behavior and parameters of the algorithm are not changed but are reorganized, with the channelSpacing, speedSound, and freq parameters are relocated now to the precomputation code, which is executed once. The precomputation speeds up the performance of the system significantly.

<u>Ultrasonix Ulterius Documentation and Installation: Ulterius source code is provided within the Ultrasonix SDK package (sdk607) on Wiki</u>

<u>To build Ulterius code on Windows:</u>

1. Download Microsoft Visual Studios (2010 version suggested by Ultrasonix documentation, but difficult to find).
2. Download CMake 2.8.8, QT4 (4.8.7), and opencv (2.4.11).
3. Run CMake 2.8.8 on sdk607 directory (create a new directory for project files)
4. Establish QT4 and opencv libraries in Visual Studios 2010.

<u>Implementation of Image Panel on Ulterius Interface: (Many thanks to Lei for providing a reference Ulterius build with QT visualization)</u>

- Modified Ulterius.ui file. Added in a viewing plane object using QT software interface on VS 2010.
- Added new options for image averaging and data saving.

In UlteriusDemo.cpp:

- Modified onNewData function (recieves data as an array from the system) to transfer array to processFrame function.
- Added processFrame function - A new function built on the UlteriusDemo.cpp. As inputs: it takes in data (1D array of chars/ints) and data type (int). Data is then transformed into a 2d array of pixels (QTarray) and send into the viewing plane object.
- Adapted PA rebeamformer into Ulterius to allow for real-time PA imaging. Made modifications to allow for image averaging and scan conversion as well.
- Added a saving option and image averaging options (in the Ulterius interface).

Effectively, the workflow for the new Ulterius interface is: Data from system –> calls onNewData function –> calls processFrame function –> display image. By adding our rebeamforming function into onNewData function (effectively convert RF array data to B mode array data) we can then send the data and proper image type into the processFrame function to display a properly beamformed image for PA signals. We can also apply envelope detection and log compression to refine the image further.