

Application Note



UTIA Ultrasound EV Board v2.0

Zdeněk Pohl, Lukáš Kohout

zdenek.pohl@utia.cas.cz, kohoutl@utia.cas.cz

Revision history

Rev.	Date	Author	Description
01	13.12.2023	Z.P.	Document creation
02	26.2.2024	Z.P.	Text corrections

Contents

1 Description.....	1
2 Board Features	2
3 How to Run.....	2
4 Description of Capture Parameters	4
5 Capture and View Recording from Individual Microphones	6
6 Capture Data in PC.....	7
7 Beamforming in PC.....	8
8 License	11
9 Content of the package.....	11
10 References	12

Acknowledgment

Under grant 101096884, Listen2Future is co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Key Digital Technologies Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them. The project is supported by the Key Digital Technologies Joint Undertaking and its members (including top-up funding by Austria, Belgium, Czechia, Germany, Netherlands, Norway, and Spain).

National Funding

This project has also received national funding from the Ministry of Education, Youth and Sports of the Czech Republic (MEYS) under grant agreement No 9A22004.

1 Description

This application note details the Ultrasound EV Board v2.0 developed by UTIA. The system comprises three hardware boards, with phased array of digital microphones designed specifically by UTIA. It includes FPGA-based hardware interfaces and the software for data capture from microphones, raw data storage, real-time data transmission to a PC via UDP, and additional software for data processing and result visualization on a PC.

The hardware platform consists of three basic components, two made by Trenez Electronic:

1. TE0821-01-2cg 4GB FPGA SoM module and,
2. TE0706-3 carrier board.

And phased microphone array PCB made by UTIA, see complete assembly in Figure 1.

The software part consists of 3 components:

1. Capture application running on hardware shown in Figure 1 and multicasting RAW captured data using UDP.
2. PC capture application capable to collect data from UDP multicast to file.
3. Beamforming PC application which operates on file containing the data and generates dataset necessary for AI training.

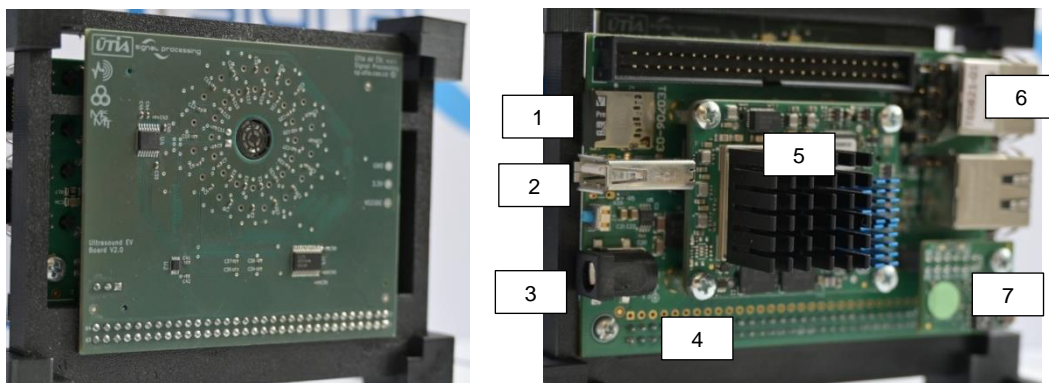


Figure 1: Complete EV board 2.0 assembly. Front side with ultrasound enabled microphone array PCB (left), rear side TE0821-1-2CG SoM on TE0706 carrier board equipped with IO interfaces (right), 1 – MicroSD card with firmware, 2 – USB 2.0, 3 – Power plug, 4 – TE0706 Carrier board, 5 – TE0821 SoM, 6 – 1Gbps Ethernet interface, 7 – JTAG/UART extension module

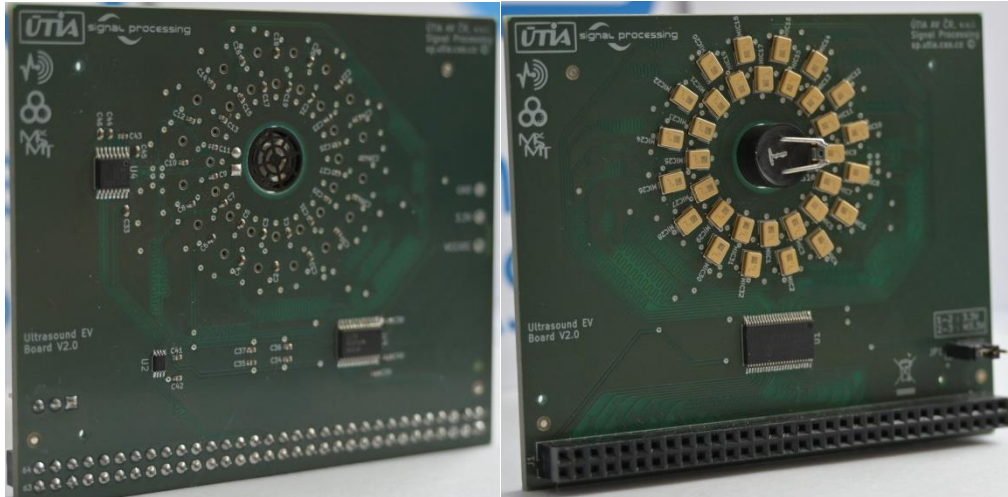


Figure 2: Ultrasound enabled digital microphone array extension board, front side with acoustic ports (left), rear side with microphones (right)

2 Board Features

The UTIA Ultrasound EV Board v2.0 implements following features:

- Microphone array consisting of 32 digital microphones IM73D122V01
- Individual microphone frequency range 0-80kHz, resonance frequency peak 36kHz
- Phased array maximal frequency is 45 kHz. Distance between microphone acoustic ports is < 3.8 mm in any direction.
- Single 40 kHz piezo US speaker with 60deg wide output beam to cover maximal area
- Common 4.8 MHz clock distribution to all microphones.
- Connected to TE0706 carrier using 2 rows of J6 header.
- Compatible with FPGA modules supporting either 3.3V or 1.8V IO.

3 How to Run

Prerequisites:

- UTIA Ultrasound EV Board v2.0 assembled with TE0706-03 carrier with TE0821-1-2CG 4GB Zynq Ultrascale+ SoM
IMPORTANT: After assembling EV Board to TE0821 carrier alignment of J6 connector pins must be checked.
- IMPORTANT:** Configuration of TE0706 Jumpers must be set as follows - see table and photo:

Part	PCB Name	Position
J10	A	2-3 short (M3.3V)
J11	B	1-2 short
J12	C	2-3 short
J13	SD	2-3 short



Figure 3: Jumper settings on carrier board TE0706-3

3. 5V power source for TE0706.
4. (optional) Mini USB cable for Virtual UART connection to PC via JTAG/UART module.
5. UTP Cable for connection to PC. We recommend using Gigabit Ethernet interface for best performance. (board uses UDP multicast to pass captured data)
6. (optional) Board stand mounted to support assembled boards in correct position.
7. The DHCP server must be running on local network where the board is connected. Alternatively, the UART connection to board can be used to setup Ethernet interface manually.
8. Micro SD card with firmware.

Updating/Writing firmware to Micro SD card:

1. Plug the Micro SD card to PC reader (it may be needed to use SD card adapter)
2. Open your favorite image writing application (Win32DiskImager for example) and write sd_card.img to the SD card.
3. Disconnect card and remove it from reader.

Running the application:

1. Insert Micro SD card with firmware to J4 slot on TE0706.
2. Plug the power source. Board should start booting automatically. If the boot process does not start automatically, the reset button S2 on carrier board TE0706 can be used to initiate booting.
3. The capture application doesn't start automatically after Linux boot. To start the application manually follow either a) or b) option:
 - a. Using serial console:
 - i. Use optional Mini USB cable and connect J4 connector on JTAG/UART module to PC.
 - ii. Use serial console application in PC (putty for example) and connect to board using parameters: 115200bps, parity none, stop bits 1, data 8bit, flow none
 - iii. (system uses autologin) in other case log in using following credentials: login: root , password: root
 - b. Using ssh server:
 - i. Find IP address assigned to board by DHCP server.
 - ii. Use ssh client on PC to connect the board.
 - iii. Use login: root , password: root

Run user application by following command:

```
> cd /mnt/sd-mmcb1k1p1
> ./capture_wave_mcast_calib.elf
```

IMPORTANT: If another instance of the application is running, it must be terminated before executing this command.

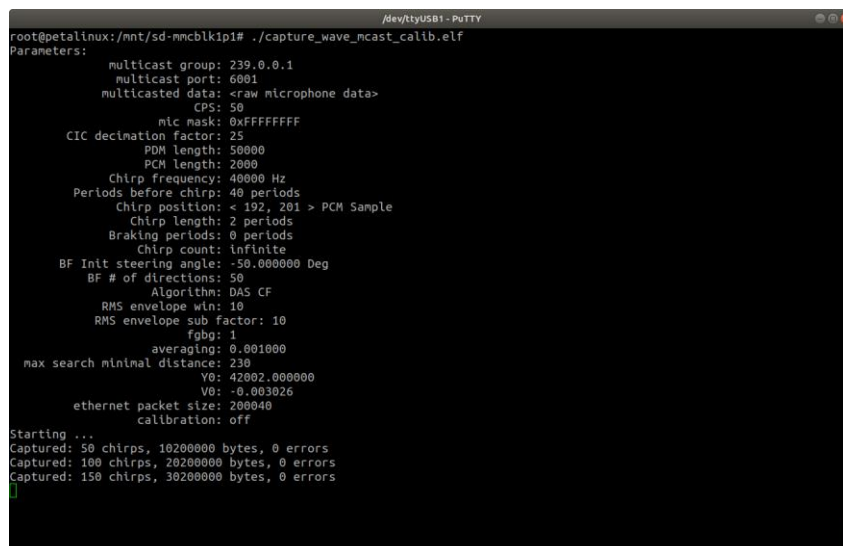
4. The board features FTP server which can be used for accessing captured data files or capture configuration file:

a. Configuration is stored in `evboard_ffbf.cfg` file at location:

```
> /mnt/sd-mmcb1k1p1/evboard_ffbf.cfg
```

5. While the application is running:

a. The actual parameters of capture process are shown in terminal:



```
root@petalinux:/mnt/sd-mmcb1k1p1# ./capture_wave_mcast_calib.elf
Parameters:
  multicast group: 239.0.0.1
  multicast port: 6001
  multicasted data: <raw microphone data>
  CPS: 50
  mic mask: 0xFFFFFFFF
  CIC declination factor: 25
  PDM length: 50000
  PCM length: 2000
  Chirp Frequency: 40000 Hz
  Periods before chirp: 40 periods
  Chirp position: < 192, 201 > PCM Sample
  Chirp length: 2 periods
  Braking periods: 0 periods
  Chirp count: infinite
  BF Init steering angle: -50.000000 Deg
  BF # of directions: 50
  Algorithm: DAS CF
  RMS envelope win: 10
  RMS envelope sub factor: 10
  fbg: 1
  averaging: 0.001000
  max search minimal distance: 230
  V0: 42002.000000
  V0: -0.003026
  ethernet packet size: 200040
  calibration: off
Starting ...
Captured: 50 chirps, 10200000 bytes, 0 errors
Captured: 100 chirps, 20200000 bytes, 0 errors
Captured: 150 chirps, 30200000 bytes, 0 errors
```

The application while running updates actual number of data recordings (chirps). The application is capable to generate US pulses and record echoes. It is also possible to use it just for recording by setting the chirp length to 0.

b. From terminal, the application can be terminated by pressing Ctrl-C.

4 Description of Capture Parameters

This section explains parameters which can modify behavior of the “capture_wave_mcast_calib.elf” application. Please note, that there are other options used in past and no longer active for this application:

Active Option	Default Value	Description
---------------	---------------	-------------

Multicast group	239.0.0.1	Multicast address where the captured data will be sent
Multicast port	6001	Multicast port
Cps request	50	How many recordings per second will be performed (see image)
Mic array mask	0xffffffff	Microphones active in capture process: bit0 - mic0, bit1 - mic1, etc.
PCM length	2000	Length of PCM samples to be recorded
Chirp Frequency	40000	Frequency of generated pulse
Lead to start	40	Number of periods before pulse will be generated
Chirp length in periods	2	Number of chirp periods
Chirp braking periods	0	Number of reverse phase periods after each chirp to brake membrane
Chirp count	0	0 - run indefinitely, any other positive - finite number of recordings/pulses to generate/capture. Last recording is always stored to file.
Calibration mode	0	0 - calibration off, normal operation 1 - calibration on, this mode is implemented for testing of newly soldered mic array boards

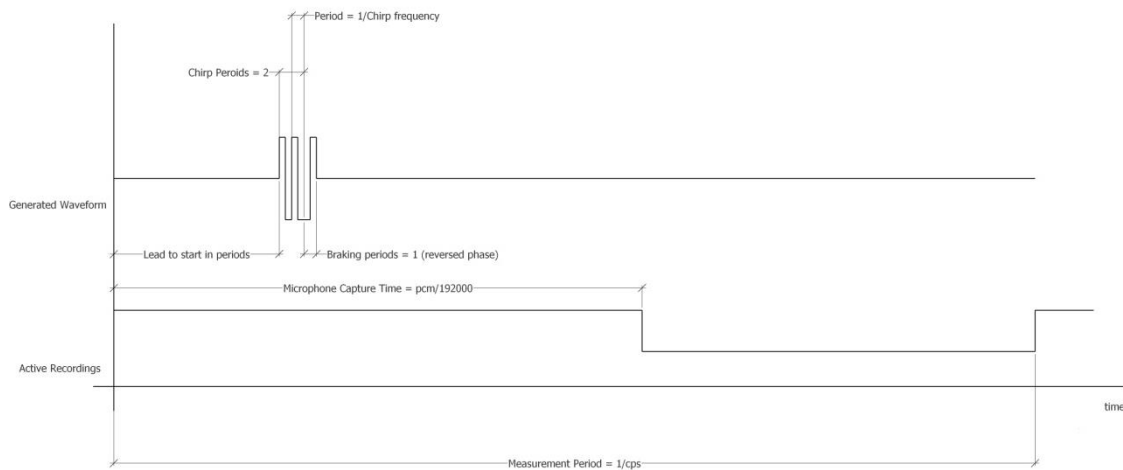


Figure 4: Periodic waveform generator and synchronous periodic recording. Waveform is parametrized by values in configuration file.

5 Capture and View Recording from Individual Microphones

1. Set parameters in `evboard_ffbf.cfg` to:
 - a. Chirp count = 1
 - b. PCM Length = required number of PCM samples at 192kHz (for example 192000 as 1 sec)
 - c. Chirp length in periods = 0 # no wave out generated
2. Run the application on the board:

```
> cd /mnt/sd-mmcb1k1p1/  
> capture_wave_mcast_calib.elf
```

3. Connect to the board using FTP from PC and copy file:

```
/mnt/sd-mmcb1k1p1/mic.raw  
to folder 'matlab'
```

4. Open Matlab in the matlab folder and run commands

```
load('utia_evboard_18_32mic_1sp.mat')  
[pcm pdm p] = load_evboard_waveforms_b('mic.raw', p_utia_evboard_18_32mic_1sp);  
mesh(pcm(2000:8000,:))
```

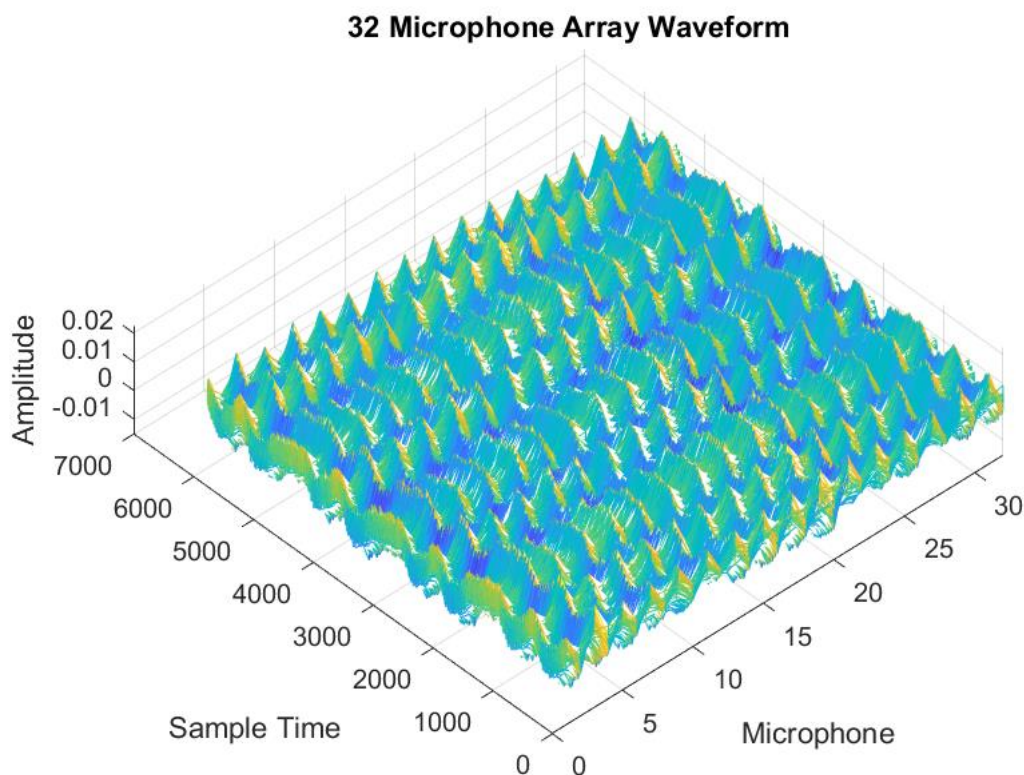


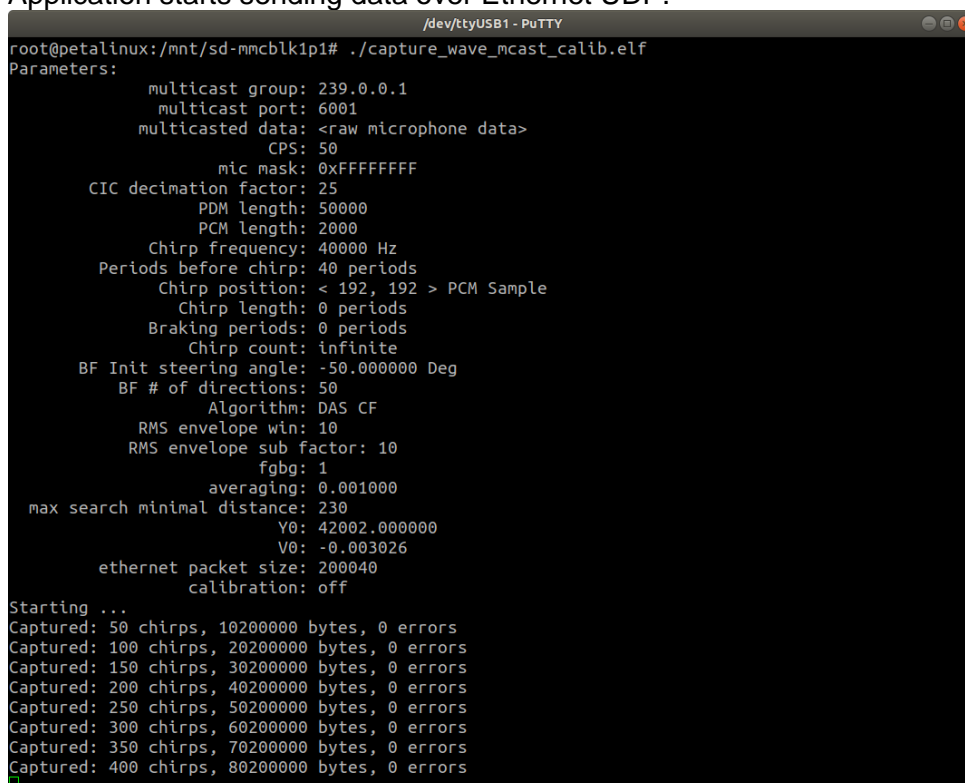
Figure 5: Example of 32 microphones recording. Signal of every other microphone has lower level as described in datasheet for microphone stereo pairs (different DC offset of microphones in stereo pair).

6 Capture Data in PC

1. Make sure that `evboard_ffbf.cfg` is in default state – for example by writing again firmware image into SD card. And modify `evboard_ffbf.cfg` values as follows:
 - a. Chirp count = 0
 - b. Chirp length in periods = 0 # no output waves will be generated
2. Connect to the board as described in “Running the application” section.
3. Run the application on the board:

```
> cd /mnt/sd-mmcblk1p1/  
> capture_wave_mcast_calib.elf
```

4. Application starts sending data over Ethernet UDP.



```
root@petalinux:/mnt/sd-mmcblk1p1# ./capture_wave_mcast_calib.elf  
Parameters:  
  multicast group: 239.0.0.1  
  multicast port: 6001  
  multicasted data: <raw microphone data>  
    CPS: 50  
    mic mask: 0xFFFFFFFF  
  CIC decimation factor: 25  
    PDM length: 50000  
    PCM length: 2000  
  Chirp frequency: 40000 Hz  
  Periods before chirp: 40 periods  
  Chirp position: < 192, 192 > PCM Sample  
  Chirp length: 0 periods  
  Braking periods: 0 periods  
  Chirp count: infinite  
  BF Init steering angle: -50.000000 Deg  
  BF # of directions: 50  
    Algorithm: DAS CF  
  RMS envelope win: 10  
  RMS envelope sub factor: 10  
    fbg: 1  
  averaging: 0.001000  
  max search minimal distance: 230  
    Y0: 42002.000000  
    V0: -0.003026  
  ethernet packet size: 200040  
  calibration: off  
  
Starting ...  
Captured: 50 chirps, 10200000 bytes, 0 errors  
Captured: 100 chirps, 20200000 bytes, 0 errors  
Captured: 150 chirps, 30200000 bytes, 0 errors  
Captured: 200 chirps, 40200000 bytes, 0 errors  
Captured: 250 chirps, 50200000 bytes, 0 errors  
Captured: 300 chirps, 60200000 bytes, 0 errors  
Captured: 350 chirps, 70200000 bytes, 0 errors  
Captured: 400 chirps, 80200000 bytes, 0 errors
```

5. Switch to Linux OS PC
6. Go to folder `bin/capture`
7. Run capture from UDP application:

```
$ ./capture_array_leap -c 100 -o array
```

IMPORTANT: This command captures 100 recordings from UDP, the result will be saved to file `array_data.raw` as specified by ‘-o’ option. The file can be used in Matlab in the same way as file ‘`mic.raw`’ in previous step. Unlike the previous case, where only one recording is stored in RAW file, in this case, there is 100 recordings of 2000 PCM sample length and recording is started with period of 1/50 sec. The number of recordings stored in RAW file is returned by “`load_evboard_waveform_b`” function in structure “`p`” as `p.fcnt`.

7 Beamforming in PC

Stored RAW data from previous section can be used to compute beamforming and visualize directions from the acoustic emissions are coming. Beamformer results are stored to file for further work. The application which can be used for this purpose is named “batch_beamforming” and the following options relevant to our data can be used to modify its behavior:

```
batch_beamforming [-v verbosity_level] [-i filename_prefix] -m mic_array_geometry -g grid_configuration
```

1. **verbosity_level** ... how many information you can get from running application
 - a. 0 (default) ... quiet run, beamformer output is written to files(s)
 - b. 1 ... 2D beamformer output preview shown while running, only one horizontal plane from beamformer is shown – the one where global maximum is located, see example image:

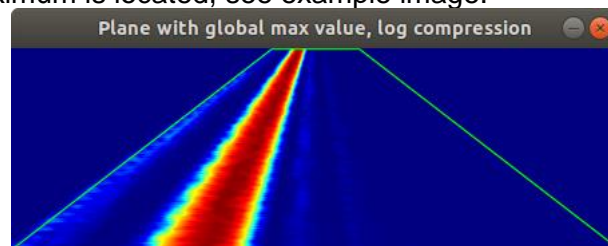


Figure 6: Example preview while running with verbosity level 1

- c. 2 ... 3D preview includes also 2D preview and operates in following steps:
 - i. First the array is shown with numbered microphones and frustum, for which beamformer response will be computed, press ‘q’ to continue:

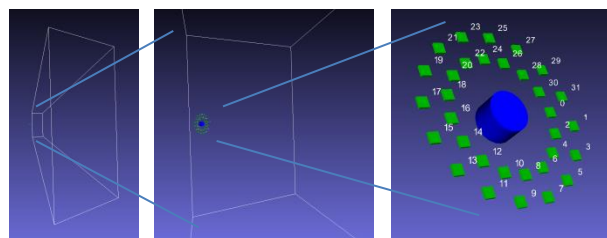


Figure 7: Verbosity level 2, 3D array and frustum visualization

- ii. Then one by one 3D renderings of each output are shown. Each window must be ended by hitting 'q' button.
WARNING: Depending on data and threshold, the visualization can be very slow; or it may fail.

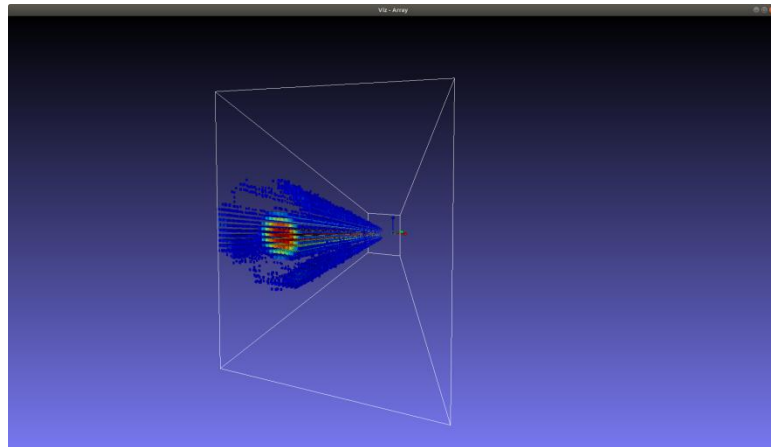


Figure 8: 3D Beamformer output visualization.

2. **filename_prefix** ... default = "array", custom part of data filename
3. **mic_array_geometry** ... CSV file containing geometry data of the array, for UTIA Ultrasound EV Board 2.0 use 'utia_ev_board_v2_0.csv'
4. **grid_configuration** ... file containing configuration of beamformer grid – the values in the file are configuring evaluated beamformer volume frustum and its sampling. The frustum is always generated symmetrical around y axis, starting by defined right-bottom-front point and sampling separately for **xz** plane and **y** axis as shown in following listing and image:

```
0      # distance type: 0 - point to point, 1 - point to plane
1      # logarithmic compression
-130.0 # starting point x coordinate
-130.0 # starting point z coordinate
-100.0 # starting point y coordinate
600.0  # y depth
5.0    # step in x and z dimensions
4.0    # step in y dimensions
```

Figure 9: Example grid configuration file

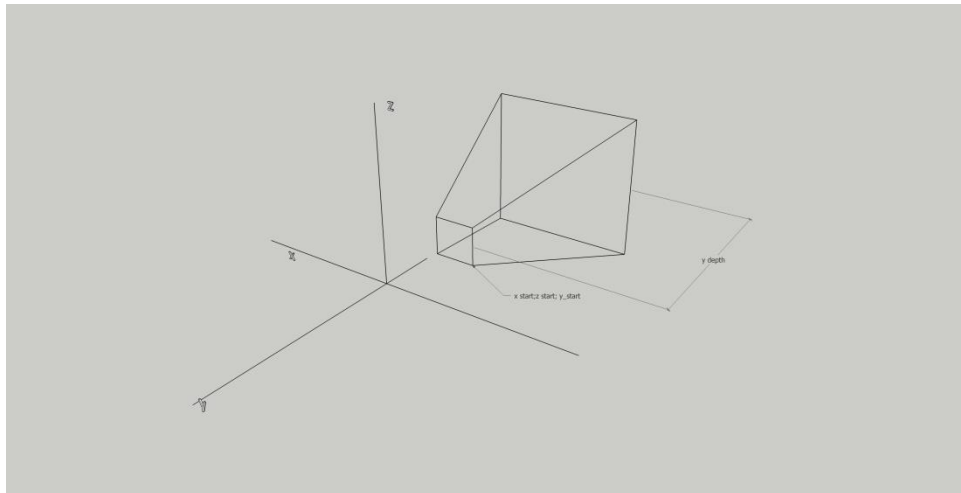


Figure 10: Frustum area for which beamformer response is computed. The 3D grid is generated from starting point, step size and symmetry around y axis assumption. Coordinates origin is the place where Mic array center is located (not shown in image)

The value 'distance type' switches method for distance computation. Value '0' uses Euclidean distance between two points, thus denoted as point-to-point. This approach allows computing near field beamformer output precisely. This option provides much better beamforming results on short distances. However, such solution is computationally intensive. We do recommend using it for distances less than 20 cm. Value '1' switches distance computation to point-to-plane distances. In that case, the beamformer assumes the source of acoustic waves is far enough from the array that the incoming waves can be assumed to be planar, thus the beamformer computation is less computationally intensive. For UTIA Ultrasound EV Board v2.0 this approach can be used for distances > 20 cm.

The value 'logarithmic compression' activates output logarithmic compression. In this case two output files are produced for each input data, with and without log compression. Log compression is better suited for visualization of output.

Output

Output of "batch_beamforming" application is stored in newly generated 'dataset' folder. It contains following files:

File	Description
data_info.txt	Contains x,y,z dimensions of output data array
train_output.txt	First column contains list of filenames numbered in order as data stored in RAW file, other columns are unused
data#.raw	3D array stored to file, dimensions are in 'data_info.txt'. The data are representing upper envelope of the acoustic signal.
logc_data#.raw	If logarithmic compression is on, the logc_ prefixed file is also generated containing the same data as data#.raw but logarithmically compressed

The data from beamformer can be also visualized in Matlab:

1. Copy file data10.raw and data_info.txt to matlab folder and run:

```

di = load('data_info.txt','-ascii');           % read dimensions
fileID = fopen('data10.raw', 'r');           % open file
numFloats = di(1)*di(2)*di(3);              % array size
data0 = fread(fileID, numFloats, 'single'); % read floats
fclose(fileID);                             % close file
rdata = reshape(data0,[di(2) di(1) di(3)]); % make 3d array
img = squeeze(sum(rdata,1));                 % make pinhole cam projection
figure,imagesc(img),colormap jet            % show the result

```

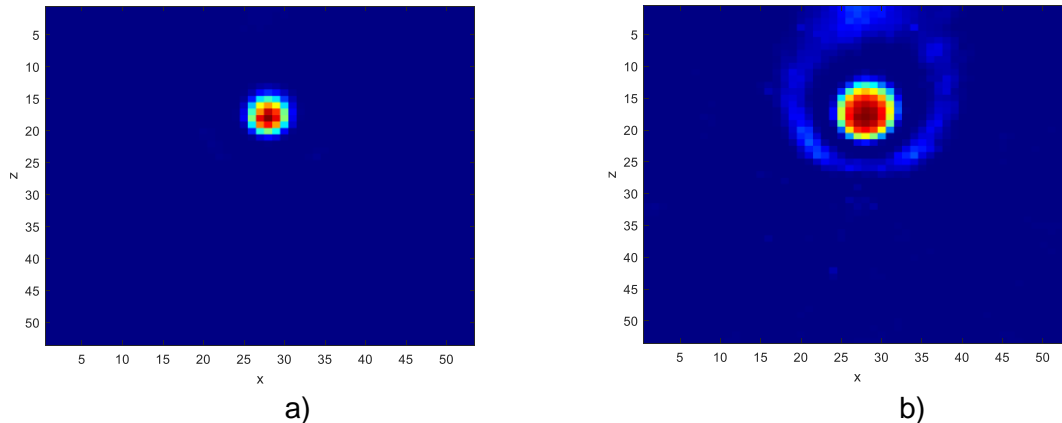


Figure 11: Example beamformer output, blob corresponds to acoustic signal source in viewfield of the array. a) log compression - off b) log compression - on. In the log compressed output, the array artefacts are more visible around the blob.

This section provides overview of beamformer HW accelerated solution of Zynq FPGA. The parameters of individual hardware cores which can be modified by the changes to configuration file are discussed.

8 License

The package is provided by UTIA AV CR as is, free of charge.

9 Content of the package

<pre> utia_ultrasound_ev_board_v20_appnote_files_v01.zip ├── bin │ ├── batch_beamforming │ ├── capture_array_leap │ ├── grid.cfg │ ├── libLeap.so │ ├── overlay_calibration.yaml │ ├── utia_ev_board_v1_x.csv │ └── utia_ev_board_v2_0.csv └── firmware </pre>	<p>Archive name</p> <ul style="list-style-type: none"> beamforming application UDP capture application Beamformer grid configuration Library required by batch_beamforming app Metadata for batch_beamforming app array configuration file for beamforming app array configuration file for beamforming app
--	--

<pre> ├── sd_card.img ├── matlab │ ├── load_evboard_waveforms_b.m │ ├── read_raw_data_b.m │ └── utia_evboard_18_32mic_1sp.mat └── UTIA_Ultrasound_EV_Board_v2_0_rev02.pdf </pre>	<p>firmware for UTIA Ultrasound EV Board v2.0</p> <p>script for loading array data utility scripts for loading data array configuration file for Matlab this document</p>
--	---

10 References

[1] Listen2Future project web pages: <https://www.listen2future.eu/home>