Application Note

UTIA Ultrasound EV Board v2.x for 3CG Platform

Zdeněk Pohl, Lukáš Kohout
Zdenek.pohl@utia.cas.cz, *kohoutl@utia.cas.cz*

Revision history

Rev.	Date	Author	Description
01	28.03.2025	Z.P., L.K.	Document creation
02			

Contents

Acknowledgement

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 Chips Joint Undertaking. Neither the European Union nor the granting authority can be held responsible for them. The project is supported by the CHIPS JU and its members (including top-up funding by Austria, Belgium, Czech Republic, 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 describes ultrasound enabled microphone array board Ultrasound EV Board v2.x assembled to TE0821-3CG Trenz SoM and TE0706-3 carrier board. The EV Board v2.x is the 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 and real-time data transmission to a PC via UDP.

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

- TE0821-01-3AE31KA(PA), Ultrascale+ ZU3CG module with 4GB DDR,
- TE0706-3 carrier board, and
- EV Board v2.0 or EV Board v2.1.

The complete device assembled from these three parts can be seen in Figure 1.

The software part consists of 2 components:

- ***capture_echo*** application running on hardware shown in Figure 1 that captures RAW data from the microphone array and multicasts them using UDP.
- ***capture*** application capable to record acoustic data to a file.

The document also describes the UDP packet to allow user built its own PC application.

2 Board Features

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

- Microphone array consisting of 32 digital microphones IM73D122V01.
- Individual microphone frequency range is 0-80 kHz, where the resonance frequency peak is near 36 kHz.
- Distance between microphone acoustic ports is less than 3.8 mm in any direction.

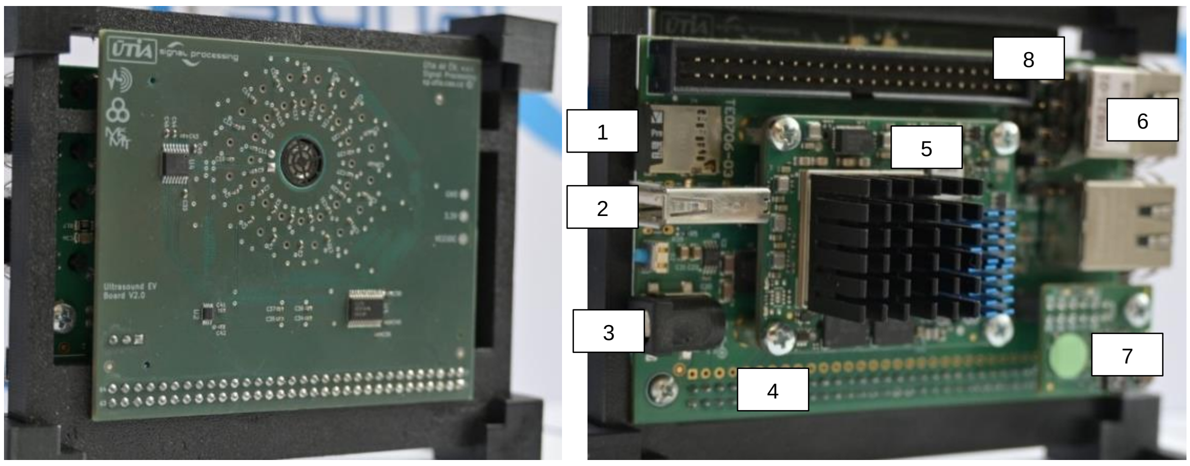


Figure 1: Complete EV Board v2.x assembly. (left) Front side with ultrasound enabled microphone array PCB, version 2.0 is shown. (right) Rear side TE0821-01-3AE31KA SoM on TE0706-3 carrier board.

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 with mini USB connector,
8 – power jumpers

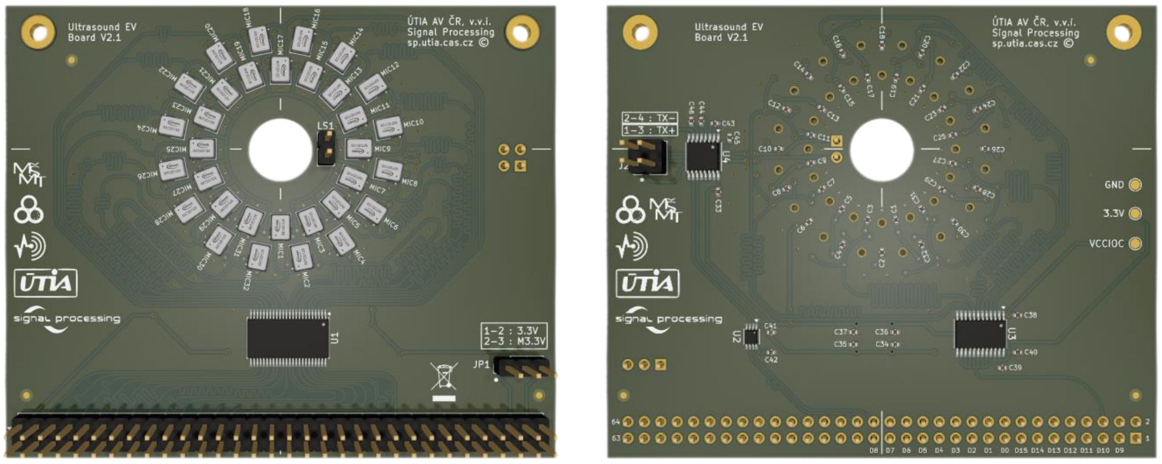


Figure 2: (left) Front side of Ultrasound EV Board v2.1 contains acoustic ports of MEMS microphones. (right) Rear side of the board with IM73D122v01 microphones.

- Single 40 kHz piezo US speaker in the middle with 60deg wide output beam.
- Microphone clock distribution supporting up to 4.8 MHz.
- The board pinout is compatible with the TE0706 carrier, it uses 2 rows of the J6 header (B and C rows).
- Compatible with 3.3V or 1.8V single ended IOs.

Figure 2 shows top and bottom side of the UTIA Ultrasound EV Board in version 2.1 which is fully interchangeable for version 2.0.

3 How to Run

Prerequisites:

1. UTIA Ultrasound EV Board v2.x assembled with TE0706-03 carrier and SoM TE0821 with UltraScale+ ZU3CG family module

   IMPORTANT: Configuration of TE0706 power jumpers must be set as follows.(see table and Figure 3)

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

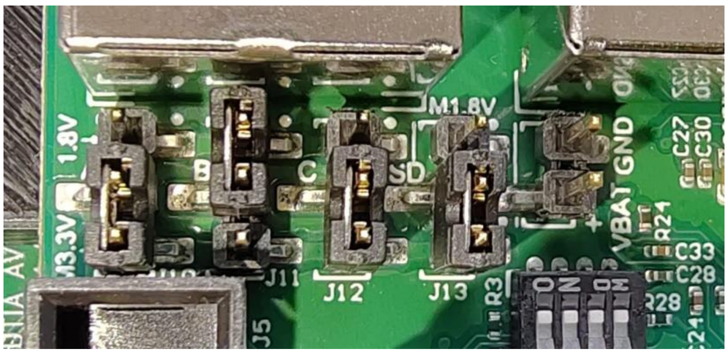


Figure 3: Jumper settings on carrier board TE0706-03, settings is applicable to both 2CG and 3CG modules.

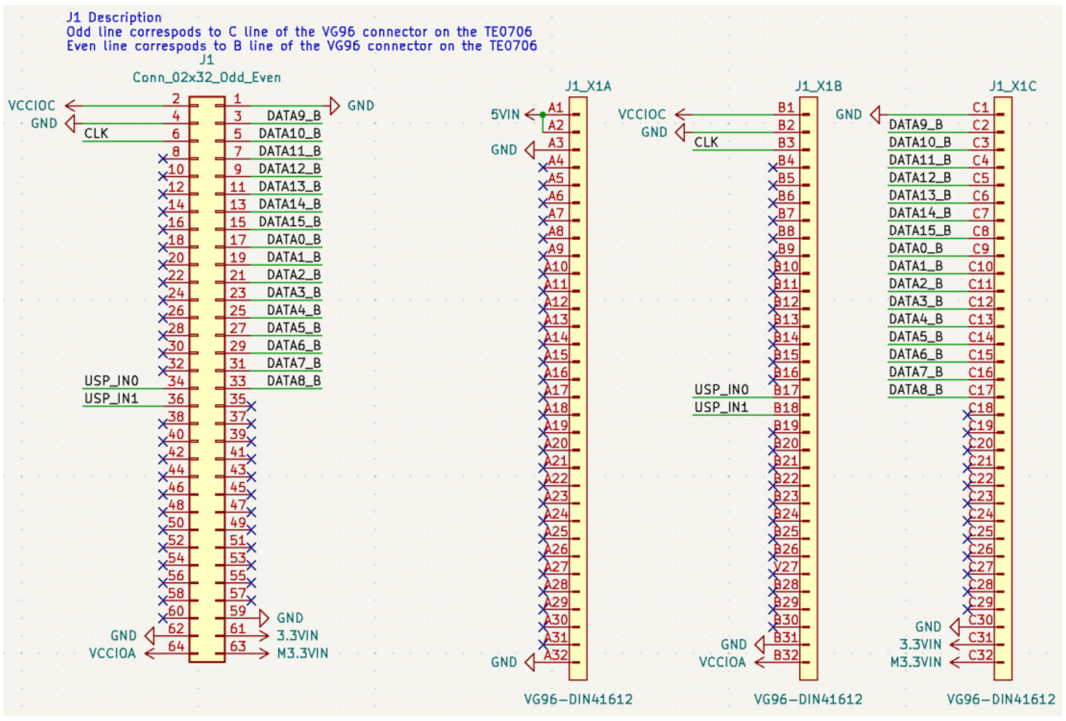


Figure 4: TE0706-03 and US EV board onnector alignment.

IMPORTANT: After assembling EV Board to TE0706 carrier the alignment of J6 connector pins must be checked. The J6 connector has three rows of pins whereas US EV board has 2 rows. Use row B and C of the J6 connector on the TE0706-03 carrier. board (see Figure 4).

2. 5V power source for TE0706.
3. (optional) Mini USB cable for Virtual UART connection to PC via JTAG/UART module (115200 bd, 8b data, 1b stop, no parity, no flow control).
4. UTP Cable for connection to PC. We recommend using Gigabit Ethernet interfaces for best performance. (the board uses UDP multicast to transmit captured data).
5. (optional) Board stand mounted to support assembled boards in correct position.
6. The DHCP server is recommended on local network where the board is connected. Alternatively, *init.sh* script on the SD card can be used to setup Ethernet interface manually.
7. 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), It is recommended to use 16 GB Micro SD card at least and it should be as fast as possible.
2. Unpack the image file *sd_card.img* from the attached ZIP file.
3. Open your favorite image writing application (Win32DiskImager for example) and write *sd_card.img* to the SD card.
4. Disconnect card and remove it from reader.
5. Plug the Micro SD card to TE0706 slot.

6. Power the board on.
7. Connect to the board using one of following options:
   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: 115200 bps, 8b data, 1b stop, no parity and no flow control.
      III. (system uses autologin) in other case log in using following credentials: login: root, password: root. Be aware that you are admin on the board.
   b. Using SSH connection:
      I. Find IP address assigned to board by DHCP server.
      II. Use ssh client on PC to connect to the board.
      III. Use login: root, password: root
8. On the running system (serial console or SSH), use the following command to extend ext4 filesystem to the remaining free space on the SD card (the ext4 partition is shrinked):

```
resize-part /dev/mmcblk1p2
```

Running the capture on the board:

The *firmware* contains two capture applications:

1. **/mnt/sd-mmcblk1p1/capture.elf** is a capture application for recording array data to a file for chosen number of seconds.
2. **/mnt/sd-mmcblk1p1/capture_echo.elf** is a capture application capable to generate US pulses and record echoes.

3.1 Application *capture.elf*

This application is intended to record data from the array for given number of seconds. Optionally, the application also collects webcam frames synchronously with the array data.

```
./capture.elf -h
Usage: ./capture.elf [-s time_in_sec -e -c cam_index -f fps -g config_file] -o fname [-v
verbosity_level -t]
  -v verbosity_level      - set verbosity level: 0 - nothing, 1 - errors, 2 - debug
  -o filename             - output filename without extension
  -s time_in_sec          - recording time in seconds, must integer > 0
  -c cam_index            - capture frames from webcam, camera index must be specified
  -d camera_calibration   - specify camera calibration XML file, if provided, frames in video
                            will be undistorted
  -f fps                  - if capture is on, set video fps rate (default is 1)
  -t                      - use external trigger (if not used, capture starts immediately)
  -g cfg_file             - use provided configuration file (if not specified,
                            evboard_ffbf.cfg file is used)
  -h                      - this Help
```

Usage examples:

Record 1 second of array data to file *test.raw*:

```
./capture.elf -o /home/root/test
```

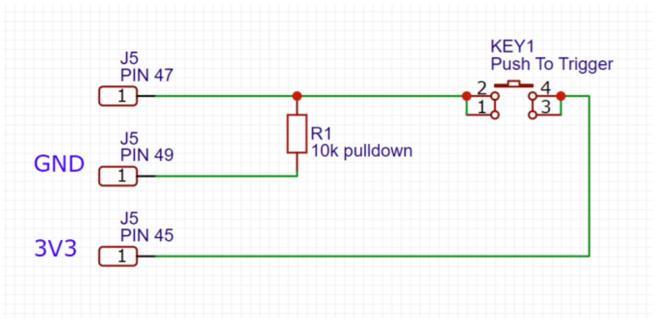


Figure 5: Example external trigger circuit.

Record 3 seconds of array data to file *test.raw*:

```
./capture.elf -s 3 -o /home/root/test
```

Record 3 seconds of array data with video to files *test_01.raw*, *test_01.avi* and first frame to *test_01.jpg*:

```
./capture.elf -s 3 -o /home/root/test_01 -c 0 -f 10
```

Triggering data capture by external signal:

```
./capture.elf -s 3 -o /home/root/test_02 -t
```

In this case of usage, capture process is initiated and stalls at the beginning until external trigger signal goes high. To use an external button as a trigger, see Figure 5.

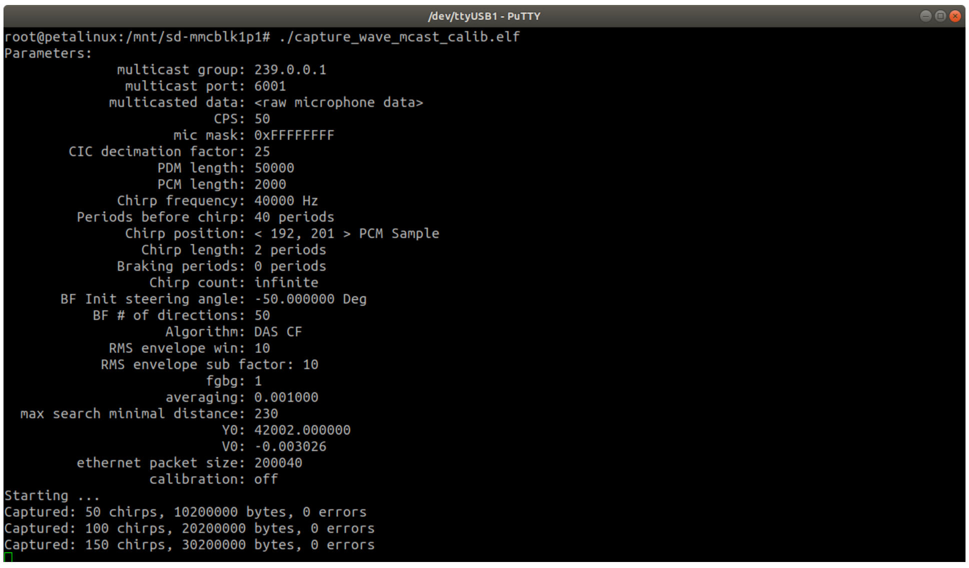
Known Problems	
Description of problem	**Solution/Workaround**
Video encoder fails to write video, error in function 'icvExtractPattern'	Output filename for video must contain '_' character inside. If not, the gstreamer uses wrong video encoder which is trying to operate on numbered images
USB driver fails to connect or communicate with webcam, suggesting 'EMI?' as a cause, consequently it fails to recover by power cycling. After that the USB port with camera remains disabled	To recover, the board must be power cycled (reset is not sufficient) To avoid this behavior, try to change power source adapter which may cause EMI to USB on TE0706 when its output is noisy
…	
…	

3.2 Application *capture_echo.elf*

This application is primarily intended to generate US pulses using its EV Board US speaker and then capture echoes of objects in front of the device. After that, the data are sent via UDP multicast group. The capturing and pulse generation are started always at the same moment. To ensure regular output pulses the hardware timer is used to trigger measurements in fixed time intervals. When the application starts, it reads its configuration from the configuration file first, and then it starts to capture and send the data.

When the application starts:

- The actual parameters of capture process are shown in terminal (Figure 6).



```
/dev/ttyUSB1 - PuTTY
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, 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
                         fgbg: 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
```

Figure 6: capture_echo.elf application terminal output.

- While the application is running it 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.
- By using '-g' command line option, the configuration file name can be specified.
- From the terminal, the application can be terminated by pressing Ctrl-C.

3.2.1 Description of *capture_echo.elf* Application Parameters

This section explains parameters which can modify behavior of the *capture_echo.elf* application. Please note, that there were also other options used in past and no longer active at this moment. Currently supported options (Figure 7):

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, PCM sampling @ 192kHz, PDM sampling @ 25*PCM
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	1	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 recoding is always stored to file 'mic.raw' for debug purposes
Calibration mode	0	0 - calibration off, must be set to 0

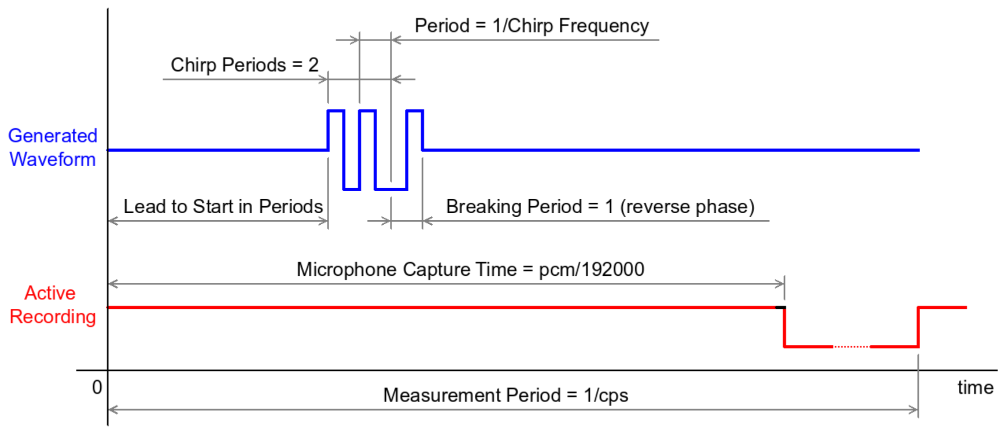

Period = 1/Chirp Frequency
Chirp Periods = 2
Generated Waveform
Lead to Start in Periods
Breaking Period = 1 (reverse phase)
Microphone Capture Time = pcm/192000
Active Recording
0
time
Measurement Period = 1/cps

Figure 7: Periodic waveform generator and synchronous periodic recording. Waveform is parametrized by values in configuration file.

3.2.2 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-mmcblk1p1/
./capture_echo.elf -g evboard_ffbf.cfg
```

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

```
/mnt/sd-mmcblk1p1/mic.raw
```

   to folder *samples* in your PC
4. Open Matlab in the *matlab* folder and run command (result can be seen in Figure 8):

```
show_me_mic_raw.m
```

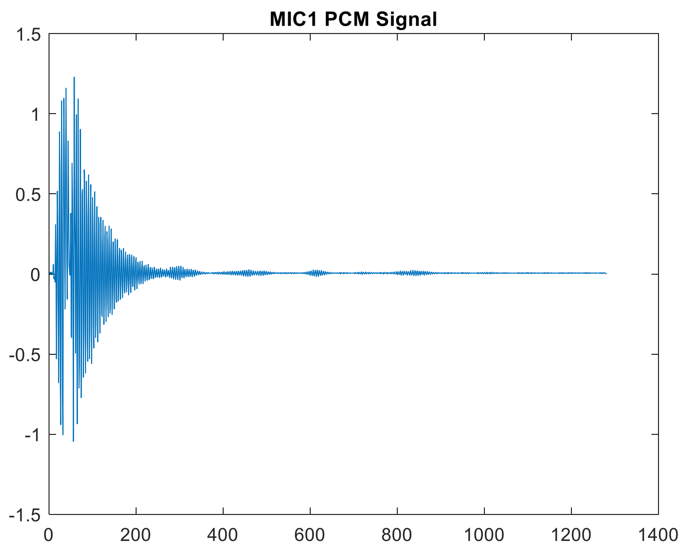


Figure 8: Example of MIC1 recording.

4 Capture Data from UDP in PC

To capture UDP data the user has to receive UDP packets from multicast and complete data from fragments. As it can be seen in Figure 7, the board generates US pulse and records echoes at the same time. The captured data from the array are then sent as UDP packets to multicast group. If the data size plus header is bigger than 1400 bytes, one packet is sent for each captured echo. Otherwise, the application will send fragments of the captured data each extended by a header. Total data size is reported by *capture_echo.elf* application after start in field *ethernet_packet_size* that is a size of one echo measurement data plus header size. The fragmentation is implemented as follows:

- Each fragment, except the last one, has size of 1400 bytes, including header size. The size of the last one is given as a size of remaining data in bytes plus size of packet header.
- Packet header structure is added at the beginning of each data fragment:

```
typedef struct packet_header{
  unsigned group_id;   // value is incremented when new
                       // measurement is sent (i.e. all
                       // fragments of one echo
                       // measurement
                       // have the same value here)
  unsigned packet_data_id; // type of data, in this case equals 2
  unsigned height;
  unsigned width;
  float u0;   //unused float value
  unsigned fps;   // number of echo measurements per second which capture_echo.elf
                  // application uses
  unsigned u1;    // unused
  unsigned offset;  // offset of this data fragment
                    // in bytes
  unsigned u2;      // unused
  unsigned chirp_delay; // delay of chirp (pulse) after
                        // start in PCM samples
} packet_header_t;
```

- Data can be received as follows:
  1. Read packet from multicast
  2. For each new **group_id** allocate memory for data of size

     height*width (in bytes)
  3. Place data from packet to correct **offset** into memory allocated for its **group_id**
  4. Data are completed if memory allocated to **group_id** is completely filled.

  NOTE: Algorithm for data receive must be robust at least to lost packets as well as the possibility that packets can arrive in different order.

5 Powering the Board Off

IMPORTANT: The board uses ext4 filesystem on second SD card partition. For that reason it cannot be turned off anytime. *poweroff* or *halt* command must be used. After the command successfully finishes, the power can be unplugged.

6 License

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

7 Content of the package

`utia_ultrasound_ev_board_v20_appnote_files_v01.zip`	Archive name
`├── firmware`	
`│   └── sd_card.img`	firmware for UTIA Ultrasound EV Board v2.x
`├── samples`	folder with data samples
`│   ├── array_02_echo`	echo recordings example
`│   │   ├── array_data.raw`	
`│   │   ├── array_video.avi`	
`│   │   └── evboard_ffbf.cfg`	
`│   ├── array_02_lab`	
`│   │   └── ca_lab_40kHz_intermittent.raw`	continuous recording of intermittent 40kHz signal
`│   ├── array_02_outside`	
`│   │   ├── ca_outside_5m_40kHz_test_01.jpg`	continuous recording of 40kHz signal, free field
`│   │   └── ca_outside_5m_40kHz_test_01.raw`	
`│   ├── mic.raw`	test data from echo recording
`│   ├── plotArray3D.m`	
`│   ├── readMicCoordinates.m`	
`│   ├── read_raw_data_b.m`	
`│   ├── show_me_array_02_echo.m`	scripts to show the captured samples
`│   ├── show_me_array_02_lab.m`	scripts to show the captured samples
`│   ├── show_me_array_02_outside.m`	scripts to show the captured samples
`│   ├── show_me_mic_raw.m`	scripts to show the captured samples
`│   └── utia_ev_board_v2_x.csv`	microphone coordinates file
`└── UTIA_Ultrasound_EV_Board_v2_x_revXX.pdf`	this document

8 References

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