Arrowhead Client on ZynqBerry Installation for Win7 and Win10

Jiri Kadlec, Lukáš Kohout
kadlec@utia.cas.cz *kohoutl@utia.cas.c*

Revision history

Rev.	Date	Author	Description
0	25.03.2019	L. Kohout	Initial version
1	1.04.2019	J. Kadlec	Installation for Win7 and Win10
2			

Contents

Acknowledgement

This work has been partially supported from project Productive4.0, project number ECSEL 737459.

1 Introduction

This application note describes an installation procedure of Arrowhead client on Zynq 7000 device. The concrete board is ZynqBerry TE0726-03M. It works with Xilinx XC77010-1C device with the dual core Arm A9 32 bit and programmable logic on single 28 nm chip. The ZybqBerry PCB has RaspberryPi 2 form factor. The ZynqBerry board is designed and manufactured by Trenz Electronic [1]. The device runs Xilinx PetaLinux 2018.2 kernel with Debian 9.8 Stretch distribution (03.25.2019). The client SW acts as a *Producer* of a service or as a *Consumer* requesting the service from an Arrowhead framework. The base hardware platform for the Zynq device is compiled with Xilinx Vivado 2018.2 tool. The entire installation procedure has been tested on Win 7 Pro and Win 10 PC. The optional configurations of the Petalinux 2018.2 kerrnel and the optional generations of the Debian image are performed in the Ubuntu 16.04 LTS host. The Ubuntu OS can be executed on the same 64bit Win7 or Win 10 PC in the VMware Workstation 14 Player. To run and test Arrowhead clients, it is required to have running Arrowhead-framework G4.0 light-weight installation running on a RaspberryPi 3B board (RPi3).

2 HW configuration with simple Arrowhead Client example

The targeted HW works with one RPi3 board and two ZynqBerry boards. The RPi3 implements the Arrowhead framework. See [2] for the documentation. The Producer ZynqBerry on the top hosts C++ provider modelling a temperature measurement. The Consumer ZynqBerry in the middle hosts C++ consumer capable to ask the Arrowhead framework about the temperature provided as service by the Producer ZynqBerry board.

Figure presents: RPi3 (bottom), and two ZynqBerry boards with the Arrowhead framework G4.0 compatible C++ clients running on ZynqBerry boards.

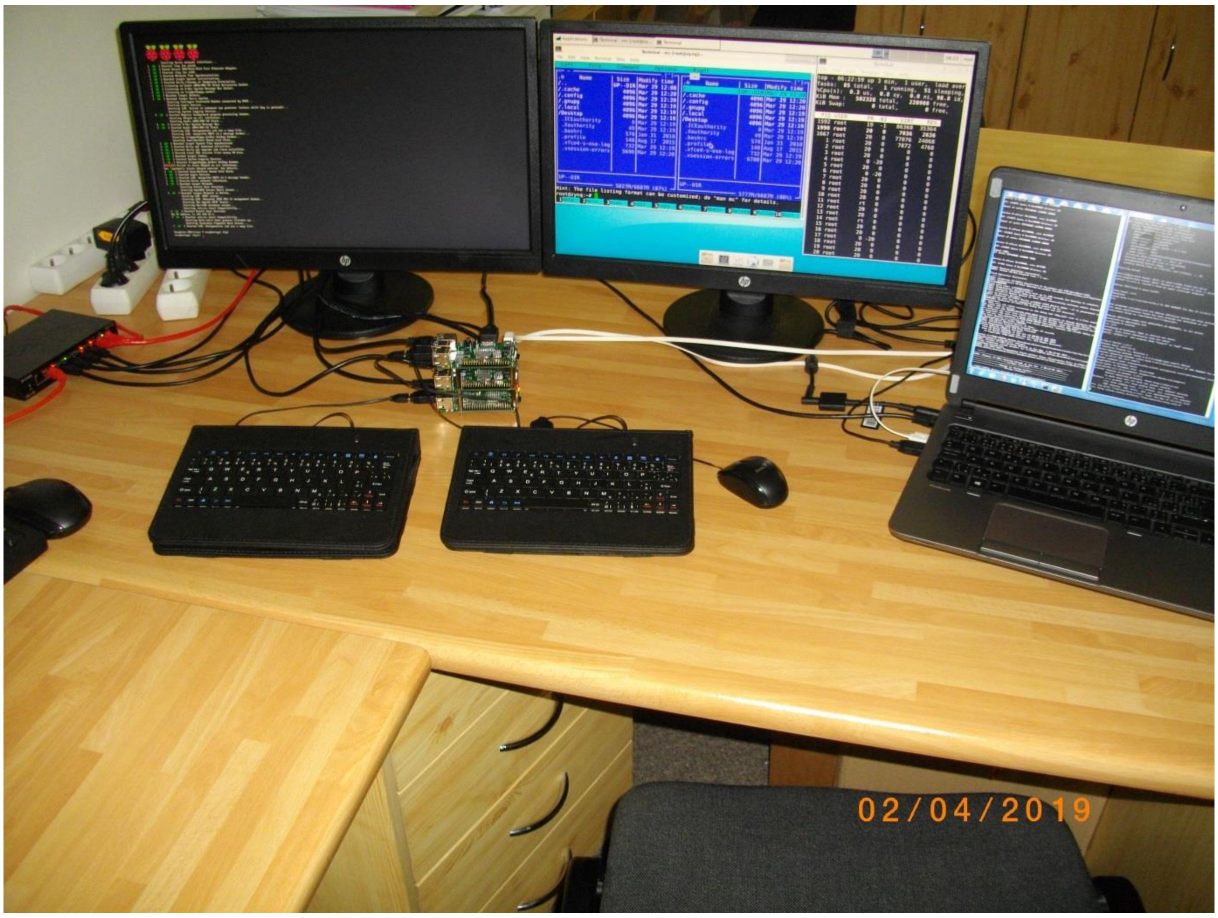


Figure presents: The initial RPi3 boot sequence is displayed on the Full HD HDMI display. ZynqBerry with USB mouse and keyboard and the Debian desktop on the HD HDMI display. The ZynqBerry HD HDMI output is supported by set of Trenz Electronic HW IPs present in the programmable logic part of the Xilinx xc7z010 device. We can see two applications running under the Debian OS on A9 processor. It is the *midnight commander* and the *top* application. Win 7 Pro PC with DOS console related to the initial programming of the Qspi FLASH of the ZynqBerry board and putty serial terminal with an initial Debian boot messages.

3 Arrowhead Services on RPi3

Testing and running of the Arrowhead C++ clients on ZynqBerry boards requires Ethernet access to the Arrowhead framework services. It is recommended to use image for the RPi3 board. It includes already installed and configured Arrowhead framework G4.0 lightweight implementation. The image is available as one of results of the work package WP1 of the running ECSEL JU project Productive4.0 https://productive40.eu/.

It is accessible for all consortium project partners from the project ownCloud repository https://productive4-cloud.automotive.oth-aw.de/index.php/login . Files are present in section WP1, task 1.4. Please contact coordinator of the consortium for further information about the access to the Arrowhead-framework G4.0 light-weight installation running on the RPi3 board. After receiving the access to the download, unzip the three downloaded files *Arrowhead-40-*

raspi.z01, *Arrowhead-40-raspi.z02* and *Arrowhead-40-raspi.zip* into the final image file *image_180626.img* (size 3.711.959.040 Bytes).

Copy the RPi3 image *image_180626.img* to (at least) 4GB SD card (speed grade 10). You can use the *Win32DiskImager* utility from: https://sourceforge.net/projects/win32diskimager/ .

Connect the RPi3 to USB keyboard, HDMI monitor with inserted SD card. Connect it to Ethernet with the DHCP server. Poweron the board by connecting the 5V power supply via micro SD cable. Power can be provided from the PC via the USB port or, preferably, from the dedicated 5V power supply.

Login as user: pi
Password: raspberry

Find and write down the assigned Ethernet IP address for IP V4 and IP V6 by typing on the RPi3 keyboard:
```
ifconfig
```

To shutdown properly the RPi3 type on the RPi3 keyboard:
```
sudo halt
```

The OS is properly shut down and all possibly open R/W to the SD card are closed. Remove temporarily the SD card and disconnect the 5V power to switch OFF the board. Return the SD card to RPi3 slot.

4 Hardware design of ZynqBerry Arrowhead client boards

The hardware is compiled with Xilinx Vivado 2018.2 tool. The design is based on a board support package provided by Trenz Electronic for the ZynqBerry board. You have to have the Xilinx Vivado 2018.2 installed on your PC. Use the “Vivado HLx 2018.2: WebPACK and Editions - Windows Self Extracting Web Installer” (EXE - 50.56 MB) for download and installation of the free Webpack version from:

https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/vivado-design-tools/2018-2.html

Use the ZynqBerry board support package version 2018.2 from Trenz Electronics to generate the HW bitstream for the programmable part of the design and the low level SW support for the preconfigured and precompiled Petalinux 2018.2 kernel and the precompiled Debian 9.8 “Stretch” image for the ZynqBerry boards. Both image files are included in this evaluation package. The complete configuration/compilation of Petalinux kernel and Debian image is skipped at this stage, but it is described in the second part of this application note (Chapter 12).

To prepare the ZynqBerry boards follow these steps:

1. Download the board support package for Xilinx tools in version 2018.2 from the Trenz Electronic web page, choose package called *ZynqBerrydemo1*: http://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/Modules_and_Module_Carriers/special/TE0726/Reference_Design/2018.2/ZynqBerrydemo1/te0726-ZynqBerrydemo1-vivado_2018.2-build_03_20181120163939.zip.
2. Unpack the package to Win 7 or Win10 directory of your choice. It will create *ZynqBerrydemo1* folder.

3. On Win 7 or Win10, open dos terminal window, go to the *ZynqBerrydemo1* folder and create an initial setup:

   cd *ZynqBerrydemo1*

```
_create_win_setup.cmd
```

   Select option (1) to create maximum setup of CMD-Files and to exit.
   Set of scripts is created in the *ZynqBerrydemo1* folder.
   To overcome limitations of Win 7 and Win10 related to the need of short directory paths, use the script _use_virtual_drive.cmd to create a virtual short path to your directory drive *X:\ ZynqBerrydemo1*   Type:

```
_use_virtual_drive.cmd
```

   Select X as name of the virtual drive and select (0) to create the virtual drive.
   Go to the created virtual sort path directory by:

```
X:
cd ZynqBerrydemo1
```

4. Use text editor of your choice and open and modify *script design_basic_settings.sh* . Select path to the Vivado 2018.2 tools Installed on your Win7 or Win10. Line 38:

```
@set XILDIR=C:/Xilinx
```

   Select proper Xilinx device. Line 48:

```
@set PARTNUMBER=3
```

   The selected number corresponds to the number defined in file
   TE0726_board_files.csv in X:\ZynqBerrydemo1\board_files/TE0726_board_files.csv file).

5. Start the Xilinx Vivado 2018.2 and create the design by executing of the script:

```
X:\ZynqBerrydemo1\vivado_create_project_guimode.sh.
```

6. Build the design, use TCL script provided within the board support package. From the Vivado TCL console execute command:

```
TE::hw_build_design -export_prebuilt
```

   After the compilation, a hardware description file *ZynqBerrydemo1.hdf* is located in:
   `X:\ZynqBerrydemo1\prebuilt\hardware\m\ZynqBerrydemo1.hdf`

   Copy the two precompiled files from this evaluation package to:

```
X:\ZynqBerrydemo1\prebuilt\os\petalinux\default\image.ub
X:\ZynqBerrydemo1\prebuilt\os\petalinux\default\u-boot.elf
```

   By copy of these two files we skip the optional Petalinux and Debian configuration and compilation steps described in Chapter 12 and 13.

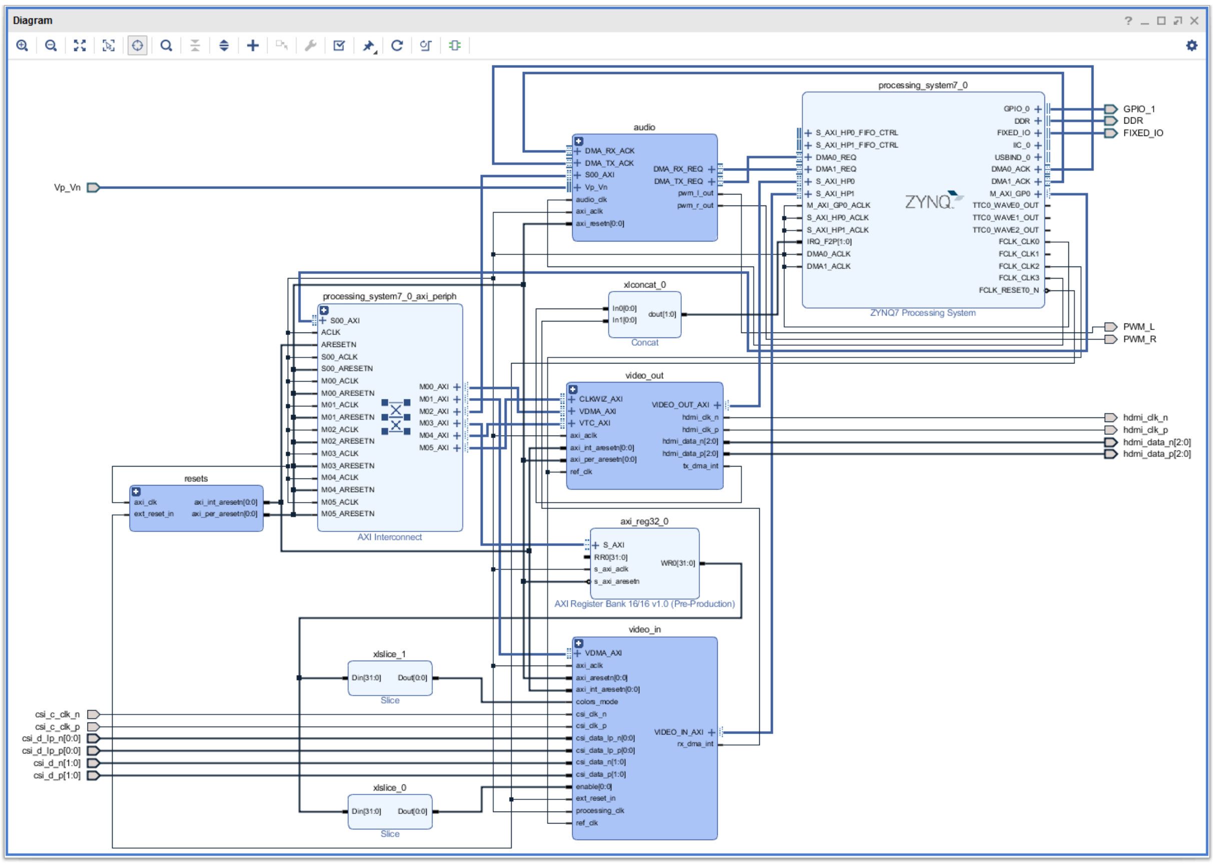


Figure shows block design of the created system. It includes HW IPs for video sensor and HW IPs for HD HDMI video output. USB controller based access to the 100 Mbit Ethernet, to 512 Mbyte DDR3 and USB ports for keyboard and mouse are pre-configured in the Zynq block.

5 Create BOOT.bin

1. In Vivado 2018.2 console, create the *BOOT.bin* file by execution of the command:
```
TE::sw_run_hsi
```
   The resulting file will be located in folder
   X:\ZynqBerrydemo1\prebuilt\boot_images\m\u-boot\BOOT.bin
   Close the Vivado 2018.2
2. Copy the created BOOT.bin file it to a newly created *NA* directory (New Application)
   X:\ZynqBerrydemo1\prebuilt\boot_images\m\NA\BOOT.bin
3. Connect the ZynqBerry board to Ethernet and power it via micro USB cable, serving also as JTAG line and also as console line.
4. From the open Win 7 or Win 10 console execute the command
```
design_clear_design_folders.cmd
```
   Scripts cleans created Vivado 2018.2 design subfolders.
5. Unzip the preconfigured and precompiled Debian image for the ZynqBerry board from from this evaluation package file: *te0726-debian.zip* (598.913.412 Bytes) to *te0726-debian.img* (7.516.192.768 Bytes).
6. Use again the *Win32DiskImager* tool for creation of the image *te0726-debian.img* on the SD card. Use 8GB SD with speed grade 10.
7. You can install and use *putty* terminal https://www.putty.org/

8. Insert created SD card to the ZynqBerry board.
9. Connect the ZynqBerry board with your Win7 or Win 10 PC via micro USB cable. The USB cable provides the 5V power supply the programming interface and console terminal. Use putty or similar terminal client with speed (baud) 115200bps, data bits 8, stop bits 1, parity none and flow control none. The used COM port number can be found in the Win7 or Win 10 Device manager too.
10. You have to write X:\ZynqBerrydemo1\prebuilt\boot_images\m\NA\BOOT.bin

    file to the ZynqBerry on-board Qspi FLASH to enable the initial stage of booting of Xilinx xc7z010 device of the ZynqBerry board. From the open Win 7 or Win 10 console execute the command

    ```
    program_flash_binfile.cmd
    ```
11. The ZynqBerry board will boot from the on board qspi flash.The first stage boot loader will load to DDR3 and start the u-boot program. The *u-boot.elf* program will start to boot the preconfigured and precompiled Petalinux *image.ub* image (size 3.926.136 bytes) from the inserted SD card with output to the *putty* serial console terminal.
12. Login the *putty* serial console terminal as

    user:

    ```
    root
    ```

    Password:

    ```
    root
    ```
13. Find and write down the assigned Ethernet IP address for IP V4 and IP V6 by typing command:

    ```
    ifconfig
    ```
14. If you have connected to the ZynqBerry board the USB mouse, USB keyboard and the HD HDMI monitor, you can start the Debian desktop by this command from the *putty* serial console terminal:

    ```
    startx&
    ```
15. To shutdown properly the ZynqBerry board type:

    ```
    sudo halt
    ```

    The Debian OS is properly shut down and all possibly open R/W to the SD card are closed. Remove temporarily the SD card and disconnect the 5V power to switch OFF the board. Return back the SD card.

Repeat steps 1-15 also for the second ZynqBerry board.

The Debian SW of both boards will be configured to become compatible with the arrowhead framework G4.0 client and provider demo application in C++. This is described in Chapters 6 to 9.

6 Install Arrowhead-f support on ZynqBerry boards

1. Start the RPi3 board, both ZynqBerry boards and Win7 or Win 10 PC.
2. Identify and write down the Ethernet addresses set by the HDCP server. The network has to support access to the external Ethernet to get access to the repositories.

   In Win7 or Win 10 PC use WinSCP or similar tool to copy the arrowhead installation script *install-arrohead-cli-dep.sh* from this evaluation package to folder of each of the two ZynqBerry boards:

```
/root/install-arrohead-cli-dep.sh
```

3. To control the ZynqBerry board, use SSH (preferred) or serial terminal of your Win7 or Win 10 PC. user: root pswd: root
4. To upgrade the Debian installation and to install dependencies required by the Arrowhead client compilation, execute from the ZynqBerry command line (SSH or serial terminal):

```
cd /root
chmod ugo+x install-arrohead-cli-dep.sh
./install-arrohead-cli-dep.sh
```

7 Install Arrowhead-f C++ Provider on ZynqBerry

To control the ZynqBerry device, use SSH (preferred) or serial terminal.

1. Get the arrowhead client source codes. The sources include C++ version of the Arrowhead *Provider* and *Client* skeletons.

```
cd /root
git clone https://github.com/arrowhead-f/client-cpp
```

2. Compile Arrowhead *Provider*.

```
cd client-cpp/ProviderExample
make
```

3. Configure the *Provider*, the name of the configuration file is *ApplicationServiceInterface.ini*.

```
mcedit ApplicationServiceInterface.ini
```

The configuration file consists of the following items.

- sr_base_uri – an address of the Arrowhead registration service running in insecure mode, in our case it is the RPi3 IP address with port 8440.
- sr_base_uri_https – an address of the Arrowhead registration service running in secure mode, in our case it is the RPi3 IP address with port 8441.
- port – a port number where the *Provider* will be available on, set 8000.
- address – *Provider* IP address, ZynqBerry IP.
- Address6 - *Provider* IP address in IPV6

The configuration file example:

```
[Server]
sr_base_uri="http://10.42.0.141:8440/serviceregistry/"
sr_base_uri_https="https://10.42.0.141:8441/serviceregistry/"
port="8000"
address="10.42.0.103"
address6="[fe80::483b:e5ff:fe7f:610d]"
```

Safe the file (F2) and exit the editor (F10).

4. Start the *Provider*

```
./ProviderExample
```

The *Provided* registers itself in the Arrowhead framework database, on *Consumer* request, it returns artificial temperature, it is fixed value 26 degrees of Celsius.

8 Install Arrowhead-f C++ Consumer on ZynqBerry

The Arrowhead *Consumer* can be compiled and run on the second ZynqBerry board. Alternatively, Consumer can be compiled and tested on the same ZynqBerry device as the *Provider*.

1. Compile Arrowhead *Consumer*.

```
cd /root/client-cpp/ConsumerExample
make
```

2. Configure the *Consumer*, there are two configuration files, *OrchestratorInterface.ini* and *consumedServices.json*.

   a. *OrchestratorInterface.ini*

   ```
   mcedit OrchestratorInterface.ini
   ```

   The configuration file consists of the following items.

   - or_base_uri – an address of the Arrowhead orchestrator service running in insecure mode, in our case it is the RPi3 IP address with port 8440.
   - sr_base_uri_https – an address of the Arrowhead orchestrator service running in secure mode, in our case it is the RPi3 IP address with port 8441.
   - port – a port number where the *Consumer* will be available on, set 8002.
   - address – *Consumer* IP address, ZynqBerry IP.
   - address6 - *Consumer* IP address in IPV6

   The configuration file example:

   ```
   [Server]
   or_base_uri="http://10.42.0.141:8440/orchestrator/orchestration"
   or_base_uri_https="https://10.42.0.141:8441/orchestrator/orchestration"
   port="8002"
   address="10.42.0.103"
   address6="[fe80::483b:e5ff:fe7f:610d]"
   ```

   Safe the file (F2) and exit the editor (F10).

   b. *consumedServices.json*

   ```
   mcedit consumedServices.json
   ```

   Modify the following items in the file:

   - requestForm/requesterSystem/port – Number of the *Consumer* port.
   - Modify line

     ```
     "security" : ""
     ```
   - preferredProviders/providerSystem/address – Preferred *Provider* IP address.
   - preferredProviders/providerSystem/port – Port number, where the preferred *Provider* listen on.

   This configuration file should look like this:

   ```
   {
     "consumerID": "TestconsumerID",
     "requestForm": {
       "requesterSystem": {
         "systemName": "client1",
         "address": "dontcare",
         "port": 8002,
         "authenticationInfo": "null"
       },
   ```

```
    "requestedService": {
      "serviceDefinition": "IndoorTemperature_ProviderExample",
      "interfaces": ["REST-JSON-SENML"],
      "serviceMetadata":{
        "security" : ""
      }
    },
    "orchestrationFlags": {
      "overrideStore" : true,
      "matchmaking" : true,
      "metadataSearch" : false,
      "pingProviders" : false,
      "onlyPreferred" : true,
      "externalServiceRequest" : false
    },
    "preferredProviders": [{
      "providerSystem":{
        "systemName": "SecureTemperatureSensor",
        "address": "10.42.0.103",
        "port":"8000"
      }
    }]
  }
}
```

Save the file (F2) and exit the mcedit editor (F10).

The Debian midnight commander tool can be started from the command line by typing:

```
mc -s
```

The two putty console programs connect via USB to the two ZynqBerry boards and display the Ethernet address automatically assigned by the DHCP server.

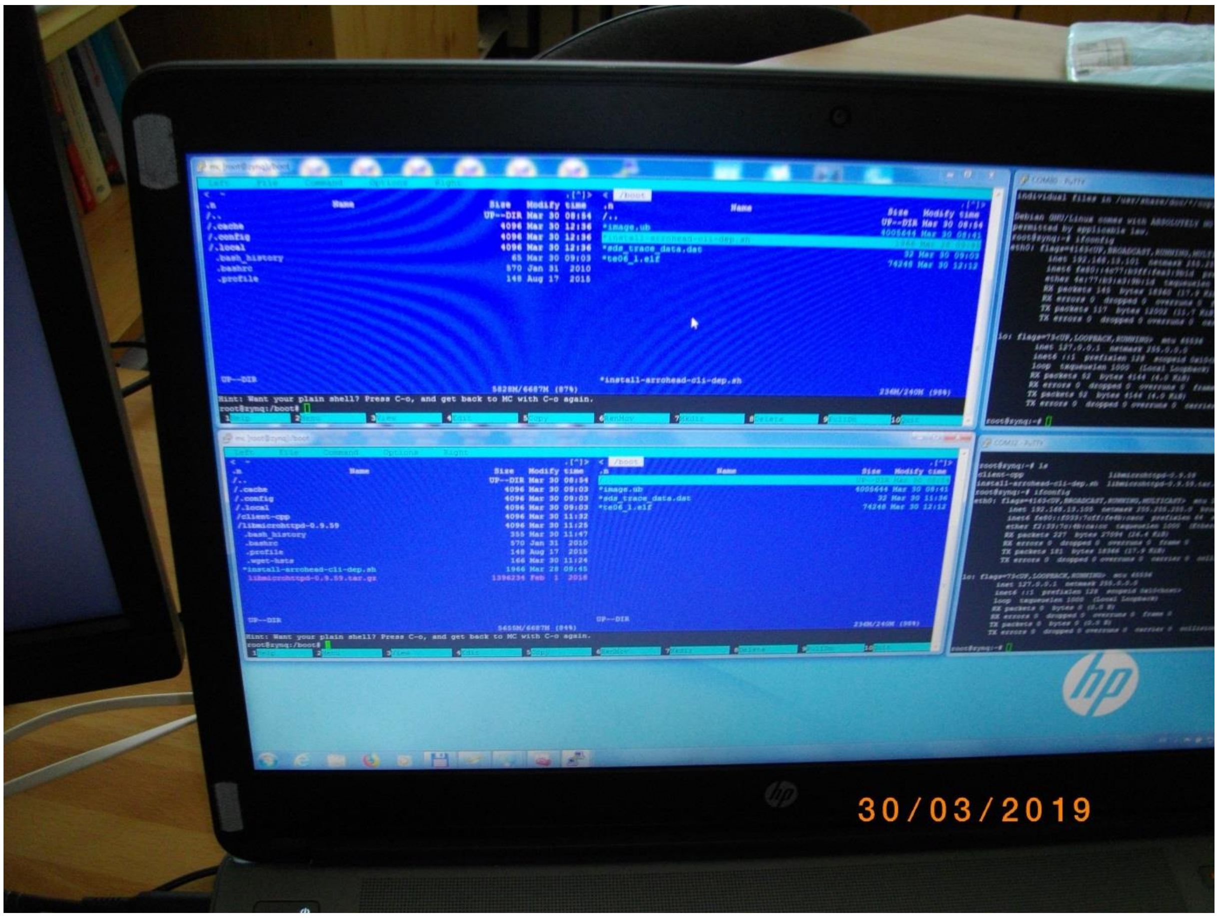
Run the *Consumer*

```
./ConsumerExample
```

The program should show the response from the *Provider* and exit.

```
Provider Response:
{"e":[{"n": "this_is_the_sensor_id","v":26.0,"t": "1553675692"}],"bn":
"this_is_the_sensor_id","bu": "Celsius"}
```

It will fail in the first instance. The database of the Arrowhead-f has to be configured. This is described in next section.



```
  Left     File     Command     Options     Right
< ~                                    .[^]>  < /boot                                      .[^]>
.n                 Name        Size  Modify time  .n                 Name          Size  Modify time
/..                          UP--DIR Mar 30 08:54 /..                            UP--DIR Mar 30 08:54
/.cache                         4096 Mar 30 12:36 *image.ub                      4005644 Mar 30 08:41
/.config                        4096 Mar 30 12:36 *install-arrohead-cli-dep.sh      1966 Mar 28 09:45
/.local                         4096 Mar 30 12:36 *sds_trace_data.dat                 32 Mar 30 09:03
 .bash_history                    65 Mar 30 09:03 *te06_1.elf                      74248 Mar 30 12:12
 .bashrc                         570 Jan 31  2010
 .profile                        148 Aug 17  2015

UP--DIR                                           *install-arrohead-cli-dep.sh
                          5828M/6687M (87%)                                      234M/240M (98%)
Hint: Want your plain shell? Press C-o, and get back to MC with C-o again.
root@zynq:/boot#
 1Help  2Menu  3View  4Edit  5Copy  6RenMov  7Mkdir  8Delete  9PullDn  10Quit
```

```
  Left     File     Command     Options     Right
< ~                                    .[^]>  < /boot                                      .[^]>
.n                 Name        Size  Modify time  .n                 Name          Size  Modify time
/..                          UP--DIR Mar 30 08:54 /..                            UP--DIR Mar 30 08:54
/.cache                         4096 Mar 30 09:03 *image.ub                      4005644 Mar 30 08:41
/.config                        4096 Mar 30 09:03 *sds_trace_data.dat                 32 Mar 30 11:36
/.local                         4096 Mar 30 09:03 *te06_1.elf                      74248 Mar 30 12:12
/client-cpp                     4096 Mar 30 11:32
/libmicrohttpd-0.9.59           4096 Mar 30 11:25
 .bash_history                   355 Mar 30 11:47
 .bashrc                         570 Jan 31  2010
 .profile                        148 Aug 17  2015
 .wget-hsts                      166 Mar 30 11:24
*install-arrohead-cli-dep.sh    1966 Mar 28 09:45
 libmicrohttpd-0.9.59.tar.gz 1396234 Feb  1  2018

UP--DIR                                           UP--DIR
                          5655M/6687M (84%)                                      234M/240M (98%)
Hint: Want your plain shell? Press C-o, and get back to MC with C-o again.
root@zynq:/boot#
 1Help  2Menu  3View  4Edit  5Copy  6RenMov  7Mkdir  8Delete  9PullDn  10Quit
```

```
individual files in /usr/share/doc/*/cop

Debian GNU/Linux comes with ABSOLUTELY NO
permitted by applicable law.
root@zynq:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTI
        inet 192.168.13.101  netmask 255.25
        inet6 fe80::4c77:b3ff:fea1:9b1d  pre
        ether 4e:77:b3:a1:9b:1d  txqueuelen
        RX packets 145  bytes 18360 (17.9 Ki
        RX errors 0  dropped 0  overruns 0
        TX packets 117  bytes 12002 (11.7 KiB
        TX errors 0  dropped 0 overruns 0  ca

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 52  bytes 4144 (4.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame
        TX packets 52  bytes 4144 (4.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier
root@zynq:~#
```

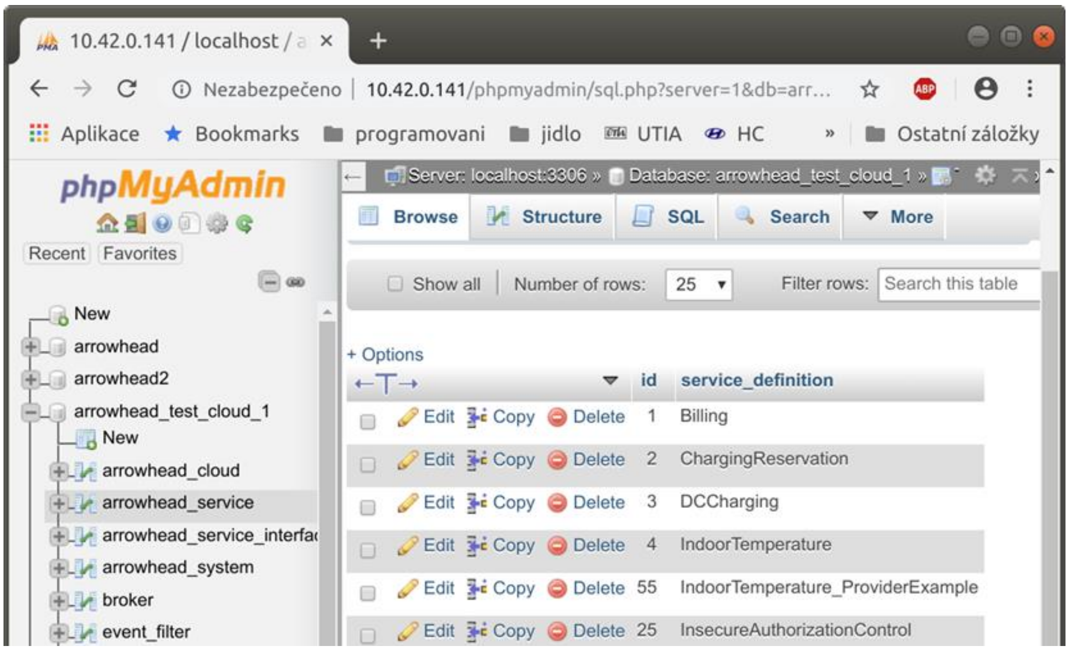
```
root@zynq:~# ls
client-cpp                   libmicrohttpd-0.9.59
install-arrohead-cli-dep.sh  libmicrohttpd-0.9.59.tar.
root@zynq:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu
        inet 192.168.13.105  netmask 255.255.255.0  bro
        inet6 fe80::f033:7cff:fe4b:ca0c  prefixlen 64
        ether f2:33:7c:4b:ca:0c  txqueuelen 1000  (Ether
        RX packets 227  bytes 27094 (26.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 181  bytes 18366 (17.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  colli

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collision
root@zynq:~#
```

Figure presents win7 laptop screen. There are with two *midnight commander* programs running on two ZynqBerry boards.

9 Modification of Arrowhead Database

The Arrowhead framework running on RPi3 provides *phpMyAdmin* to control its database. To allow the *Consumer* to get the *Producer* service response, follow next steps.


10.42.0.141 / localhost / a
Nezabezpečeno | 10.42.0.141/phpmyadmin/sql.php?server=1&db=arr...
Aplikace Bookmarks programovani jidlo UTIA HC Ostatní záložky
phpMyAdmin
Recent Favorites
New
arrowhead
arrowhead2
arrowhead_test_cloud_1
New
arrowhead_cloud
arrowhead_service
arrowhead_service_interfac
arrowhead_system
broker
event_filter
Server: localhost:3306 » Database: arrowhead_test_cloud_1 »
Browse Structure SQL Search More
Show all Number of rows: 25 Filter rows: Search this table
+ Options

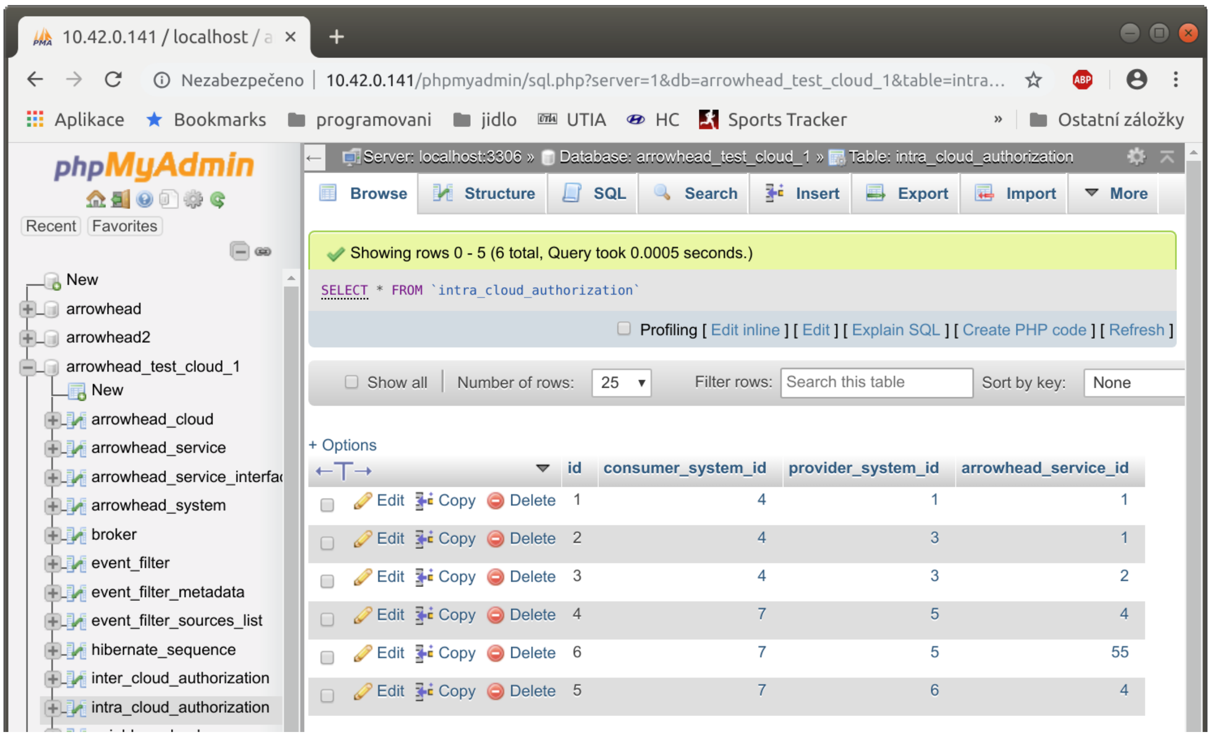
				id	service_definition
	Edit	Copy	Delete	1	Billing
	Edit	Copy	Delete	2	ChargingReservation
	Edit	Copy	Delete	3	DCCharging
	Edit	Copy	Delete	4	IndoorTemperature
	Edit	Copy	Delete	55	IndoorTemperature_ProviderExample
	Edit	Copy	Delete	25	InsecureAuthorizationControl

1. On your Win7 or Win 10 PC, start web browser and go to RPi3 *phpMyAdmin* web page, *http://10.42.0.141/phpmyadmin* (use IP address of your RPi3). User name is *root*, password is *root*.
2. Get an ID of the *Producer*. Select table *arrowhead_test_cloud_1→arrowhead_system* and locate the line containing the IP address of the ZynqBerry with system_name *SecureTemperatureSensor*.

   In our case the ID is 5.
3. Get an ID of the *Consumer*.
   Select table *arrowhead_test_cloud_1→ arrowhead_system*
   and locate the line containing system_name:
   client1.
   In our case it is 7.
4. Get an ID of the *Producer* service.
   Select table *arrowhead_test_cloud_1→ arrowhead_service*
   and locate the line containing service_definition called:
   IndoorTemperature_ProviderExample.
   In our case the ID is 55.

   Check in table *service_registry* if the *Provider* is linked with its service.
   Link the *Provider*, its service and the *Consumer* together. In table
   intra_cloud_authorization, add a new line containing: *consumer_system_id* 7,
   provider_system_id 5 and *arrowhead_service_id 55*.

The *Consumer* should get the proper response from the *Provider, now*.



10.42.0.141 / localhost

Nezabezpečeno | 10.42.0.141/phpmyadmin/sql.php?server=1&db=arrowhead_test_cloud_1&table=intra...

Server: localhost:3306 » Database: arrowhead_test_cloud_1 » Table: intra_cloud_authorization

Showing rows 0 - 5 (6 total, Query took 0.0005 seconds.)

```
SELECT * FROM `intra_cloud_authorization`
```

id	consumer_system_id	provider_system_id	arrowhead_service_id
1	4	1	1
2	4	3	1
3	4	3	2
4	7	5	4
6	7	5	55
5	7	6	4

10 Test the ZynqBerry Consumer and Producer

The *ProducerExample* server is running on the Producer ZynqBerry board.

Execute the *ConsumerExample* client example on the Consumer ZynqBerry board.

```
./ConsumerExample
```

The *ConsumerExample* client example program should show the response from the *Provider* and exit.

```
Provider Response:
{"e":[{"n": "this_is_the_sensor_id","v":26.0,"t": "1553675692"}],"bn":
"this_is_the_sensor_id","bu": "Celsius"}
```

This concludes the complete demo of Producer and Consumer on two ZynqBerry boards omplemented as C++ SW code compatible with the Arrowhead framework G4.0 lite-installation on the RPi3 board.

Producer service and Consumer client can run on a single Zynqbeery board or two different Zynqbeery boards. The configuration files and the configuration of the Arrowhead framework database described in Chapter 6 - Chapter 10 provides setup for single ZynqBerry board.

Change of the setup for two ZynqBerry boards involves only modification of the corresponding Ethernet addresses as assigned by the DHCP server.

11 Debian graphical desktop support on Zynaberry (optional)

The configuration of both ZynqBerry boards supports USB keyboard, USB mouse and the HD HDMI display desktop.

1. Start the RPi3, both ZynqBerry boards with USB keyboard, USB mouse and HD HDMI monitor and the Win7 or Win 10 PC. Identify the Ethernet address set by the HDCP server. The DHCP server has to support access to the external Ethernet.
2. To control each ZynqBerry board, use SSH (preferred) or serial terminal of your Win7 or Win 10 PC.
3. To start the desktop, type from the PC console:
   ```
   startx&
   ```
   The Debian desktop will start.

The HD HDMI display is supported by the HW interface present in the programmable part of Xilinx xc7z010 device of the ZynqBerry board.

12 Configuration of PetaLinux and Debian (optional)

The configuration/compilation of the Petalinux 2018.2 kernel and Debian 9.8 Stretch image is described in the second part of this application note.

The configuration/compilation requires Ubuntu 2016-04 installation on 64bit virtual machine. We have used as virtual machine the *VMware Workstation 14 Player* on Win7 or Win10 PC based on the Intel i7 CPU (8 processors, 16 GB RAM).

For the Ubuntu image we use the configuration of the VM machine with allocated 6 processors and 8 processors and 8 GB of RAM.

The Petalinux 2018.2 distribution is downloaded to the Ubuntu 2016-04 from

https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/embedded-design-tools/2018-2.html

and installed to the default Ubuntu directory:

```
/opt/petalinux/petalinux-v2018.2-final/settings.sh
```

To target the Debian OS, and the PetaLinux 2018.2 distribution provided by the Trenz Electronic requires modification helping to configure the Petalinux kernel image and its file system on two separate partitions of the SD card.

1. On PC Win7 or Win10 execute all steps as described in chapter 3.
2. Copy to the Ubuntu OS all content of these to Win7 or Win 10 directories:
   ```
   X:\ZynqBerrydemo1\prebuilt
   X:\ZynqBerrydemo1\os
   ```
   to Ubuntu directories:
   ```
   /home/devel/work/TV0726/ZynqBerrydemo1/os
   /home/devel/work/TV0726//ZynqBerrydemo1/prebuilt
   ```
   Copy the Debian configuration script *install-arrohead-cli-dep.sh* from this evaluation package to
   ```
   cd /home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux/ install-arrohead-cli-dep.sh
   ```
3. In Ubuntu, open linux terminal window and set path to PetaLinux 2018.2 tool (modify the path if necessary):
   ```
   source /opt/petalinux/petalinux-v2018.2-final/settings.sh
   ```

4. Go to the folder with PetaLinux, it already contains a prepared configuration according to ZynqBerry board requirements.

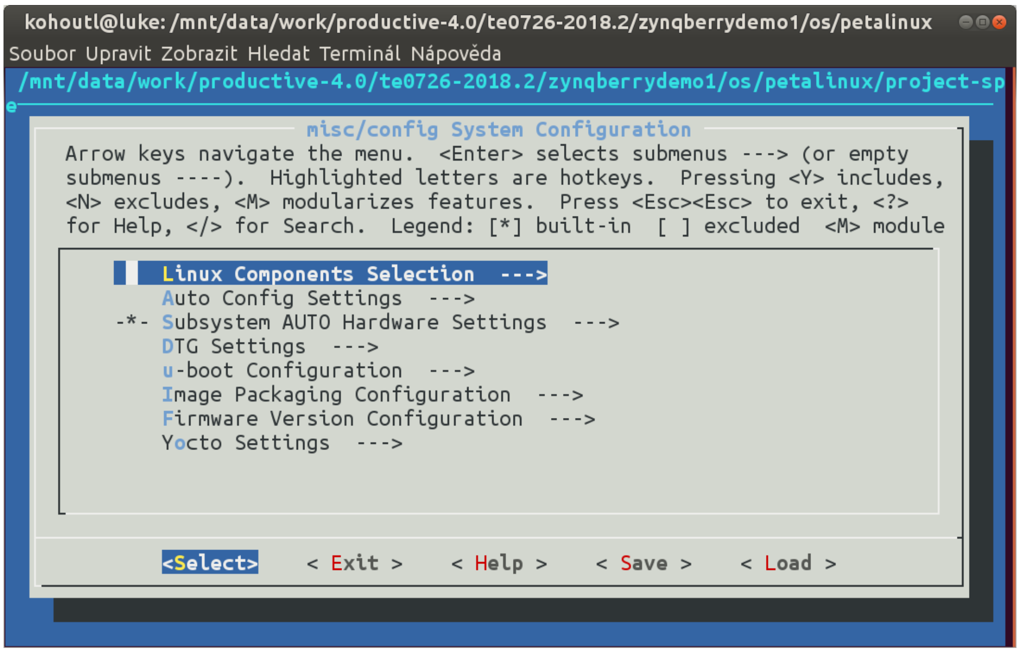
```
cd /home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux
```

5. The HDF file created (see chapter 3) in Win7 or Win 10 in Vivado 2018.2 tool is therefore in the Ubuntu folder:

```
/home/devel/work/TV0726/ZynqBerrydemo1/prebuilt/hardware/m/ZynqBerrydemo1
.hdf
```

6. Load the HDF to current PetaLinux configuration.

```
petalinux-config
--get-hw-description=/home/devel/work/TV82/prebuilt/hardware/m
```



```
kohoutl@luke: /mnt/data/work/productive-4.0/te0726-2018.2/zynqberrydemo1/os/petalinux
Soubor Upravit Zobrazit Hledat Terminál Nápověda
/mnt/data/work/productive-4.0/te0726-2018.2/zynqberrydemo1/os/petalinux/project-sp
e
                    misc/config System Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> includes,
  <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?>
  for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module

          Linux Components Selection  --->
          Auto Config Settings  --->
      -*- Subsystem AUTO Hardware Settings  --->
          DTG Settings  --->
          u-boot Configuration  --->
          Image Packaging Configuration  --->
          Firmware Version Configuration  --->
          Yocto Settings  --->

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

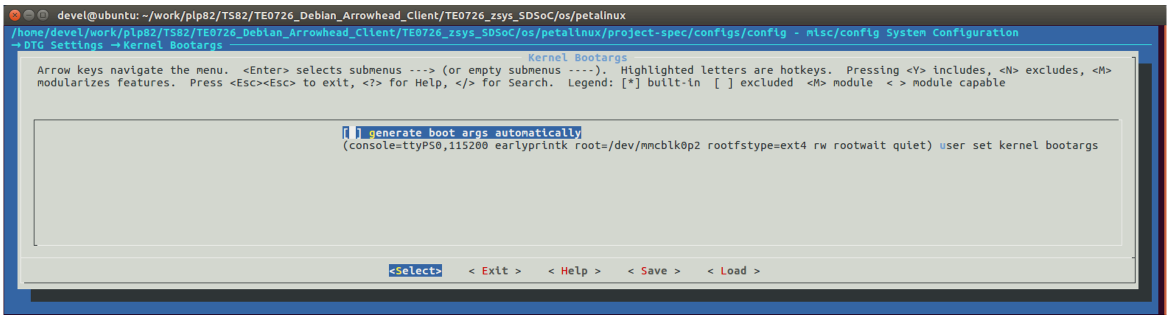
7. Change the PetaLinux filesystem location from the ramdisk to the extra partition on the SD card, select:

```
Image Packaging Configuration  --->
      Root filesystem type (SD card)  --->.
```

8. Disable option generate boot args automatically and set user defined arguments to

```
console=ttyPS0,115200 earlyprintk root=/dev/mmcblk0p2 rootfstype=ext4 rw
rootwait quiet
```

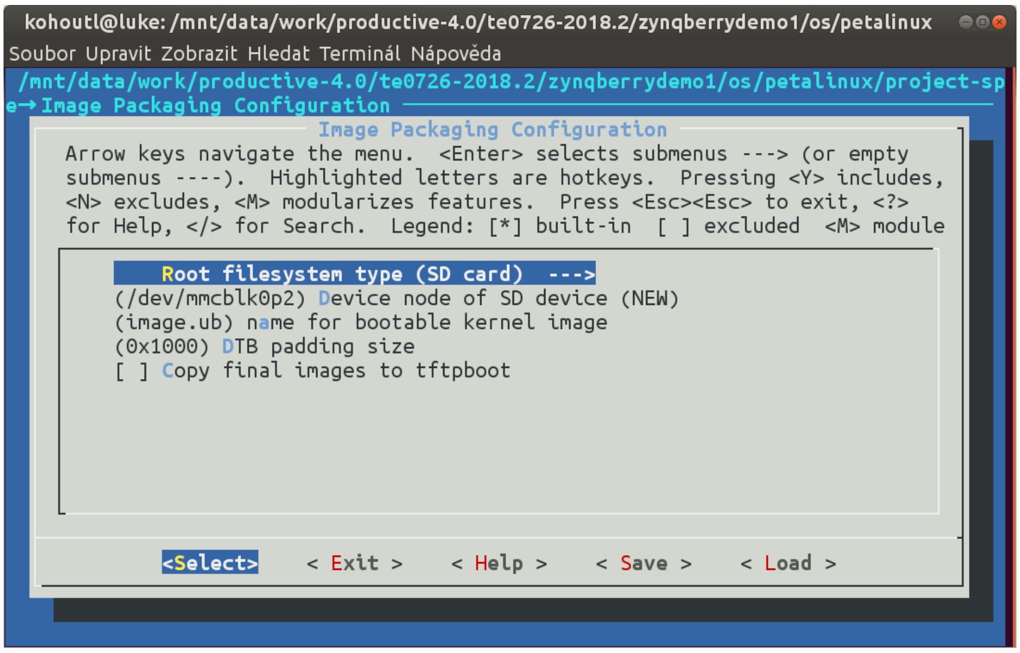
Leave the configuration, 3x *Exit* and *Yes*.



```
devel@ubuntu: ~/work/plp82/TS82/TE0726_Debian_Arrowhead_Client/TE0726_zsys_SDSoC/os/petalinux
/home/devel/work/plp82/TS82/TE0726_Debian_Arrowhead_Client/TE0726_zsys_SDSoC/os/petalinux/project-spec/configs/config - misc/config System Configuration
→ DTG Settings → Kernel Bootargs
                                        Kernel Bootargs
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M>
  modularizes features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable

                        [ ] generate boot args automatically
                        (console=ttyPS0,115200 earlyprintk root=/dev/mmcblk0p2 rootfstype=ext4 rw rootwait quiet) user set kernel bootargs

                        <Select>    < Exit >    < Help >    < Save >    < Load >
```


```
kohoutl@luke: /mnt/data/work/productive-4.0/te0726-2018.2/zynqberrydemo1/os/petalinux
Soubor Upravit Zobrazit Hledat Terminál Nápověda
 /mnt/data/work/productive-4.0/te0726-2018.2/zynqberrydemo1/os/petalinux/project-sp
e→ Image Packaging Configuration
                       Image Packaging Configuration
    Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
    submenus ----).  Highlighted letters are hotkeys.  Pressing <Y> includes,
    <N> excludes, <M> modularizes features.  Press <Esc><Esc> to exit, <?>
    for Help, </> for Search.  Legend: [*] built-in  [ ] excluded  <M> module

            Root filesystem type (SD card)  --->
        (/dev/mmcblk0p2) Device node of SD device (NEW)
        (image.ub) name for bootable kernel image
        (0x1000) DTB padding size
        [ ] Copy final images to tftpboot

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

9. Build PetaLinux, from the bash terminal execute

```
petalinux-build
```

10. Files *image.ub* and *u-boot.elf* are created in Ubuntu folder

```
/home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux/images/linux/image.ub
/home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux/images/linux/u-boot.elf
```

13 Configuration and compilation of Debian for ARM (optional)

The file system is based on the latest stable version of Debian 9.8 Stretch distribution (03. 25. 2019). Follow the steps below.

1. Copy the *mkdebian.sh* file from this evaluation package distribution to the PetaLinux folder.

```
/home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux/mkdebian.sh
```

2. Go to the folder with PetaLinux:

```
cd /home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux
```

3. Debian image will be created by execution of the *mkdebian.sh* script. The script checks all the tools that are needed to create the image, most of them are a standard part of the Ubuntu 16.04 LTS distribution. When some of them are missing, install them.

```
sudo apt install package_of_the_missing_tool
```

Next table summarizes all the tools with a corresponding package name.

Tool	Package
dd	coreutils
losetup	mount
parted	parted
lsblk	util-linux
mkfs.vfat	dosfstools
mkfs.ext4	e2fsprogs
debootstrap	debootstrap
gzip	gzip
cpio	cpio
chroot	coreutils
apt-get	apt
dpkg-reconfigure	debconf
sed	sed
locale-gen	locales
update-locale	locales
qemu-arm-static	qemu-user-static

4. Create the image with Debian, it will consist of two partitions.

   The file system of the first one will be FAT32, this partition is dedicated for image of the PetaLinux kernel.

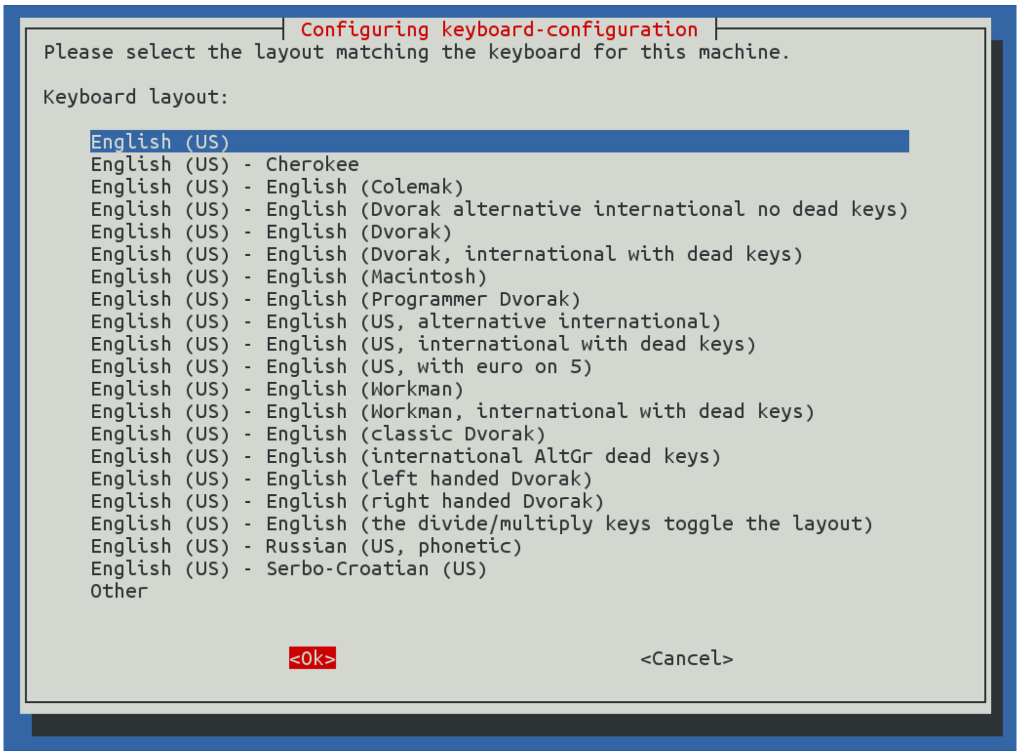
   The second partition will contain the Debian using EXT4 file system.

   Create the Debian image from the external Ethernet repositories by this command:

```
chmod ugo+x mkdebian.sh
sudo ./mkdebian.sh
```

   During the creation procedure, you will be asked to set language, choose *English (US)*. The resultant image file will be called *te0726-debian.img*, its size will be 7 GB.

   This step can take some time. It depends on the host machine speed and speed of the internet connection. Precompiled image can be found in the *te0726-arrohead-client/debian/te0726-debian.img.zip* file.



```
Configuring keyboard-configuration
Please select the layout matching the keyboard for this machine.

Keyboard layout:

    English (US)
    English (US) - Cherokee
    English (US) - English (Colemak)
    English (US) - English (Dvorak alternative international no dead keys)
    English (US) - English (Dvorak)
    English (US) - English (Dvorak, international with dead keys)
    English (US) - English (Macintosh)
    English (US) - English (Programmer Dvorak)
    English (US) - English (US, alternative international)
    English (US) - English (US, international with dead keys)
    English (US) - English (US, with euro on 5)
    English (US) - English (Workman)
    English (US) - English (Workman, international with dead keys)
    English (US) - English (classic Dvorak)
    English (US) - English (international AltGr dead keys)
    English (US) - English (left handed Dvorak)
    English (US) - English (right handed Dvorak)
    English (US) - English (the divide/multiply keys toggle the layout)
    English (US) - Russian (US, phonetic)
    English (US) - Serbo-Croatian (US)
    Other

                <Ok>                          <Cancel>
```

5. Compress the created image to file te0726-debian.zip:

```
zip te0726-debian te0726-debian.img
```

6. Copy from Ubuntu

```
/home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux/te0726-debian.zip
```

to Win7 or Win 10 file:

```
X:\ZynqBerrydemo1\prebuilt\os\petalinux\default\te0726-debian.zip
```

7. Copy from Ubuntu

```
/home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux/images/linux/image.ub
```

to Win7 or Win 10 file:

```
X:\ZynqBerrydemo1\prebuilt\os\petalinux\default\image.ub
```

8. Copy from Ubuntu

```
/home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux/images/linux/u-boot.elf
```

to Win7 or Win 10 file:

```
X:\ZynqBerrydemo1\prebuilt\os\petalinux\default\u-boot.elf
```

9. Clean Petalinux project files

```
petalinux-build -x mrproper
```

10. Delete files

```
/home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux/te0726-debian.zip
/home/devel/work/TV0726/ZynqBerrydemo1/os/petalinux/te0726-debian.img
```

11. Close the Ubuntu 2016-04 operating system.
12. In Win7 or Win 10, close VMware Workstation Player 14.

You can continue with preparation of the ZynqBerry board (as described in chapter 5) and use these newly created configured and compiled files:

- Petalinux kernel image *image.ub*
- Compressed Debian image *te0726-debian.zip*
- U-boot program *u-boot.elf*

This ends the optional configuration and compilation step for the Petalinux and Debian.

14 Package content

```
├── debian
│   ├── mkdebian.sh
│   ├── image.ub
│   ├── u-boot.elf
│   └── te0726-debian.zip
└── zynq
    └── install-arrohead-cli-dep.sh
```

References

[1 Trenz Electronic, "TE0726 TRM," [Online]. Available:https://wiki.trenz-electronic.de/display/PD/TE0726+TRM .

[2]] Documents for Arrowhead Framework Available:https://forge.soa4d.org/docman/?group_id=58

Disclaimer