

UltraZed-EG Starter Kit PetaLinux Integrated Sensors Reference Design

© 2017 Avnet. All rights reserved. All trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Avnet is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Avnet makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Avnet expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Introduction

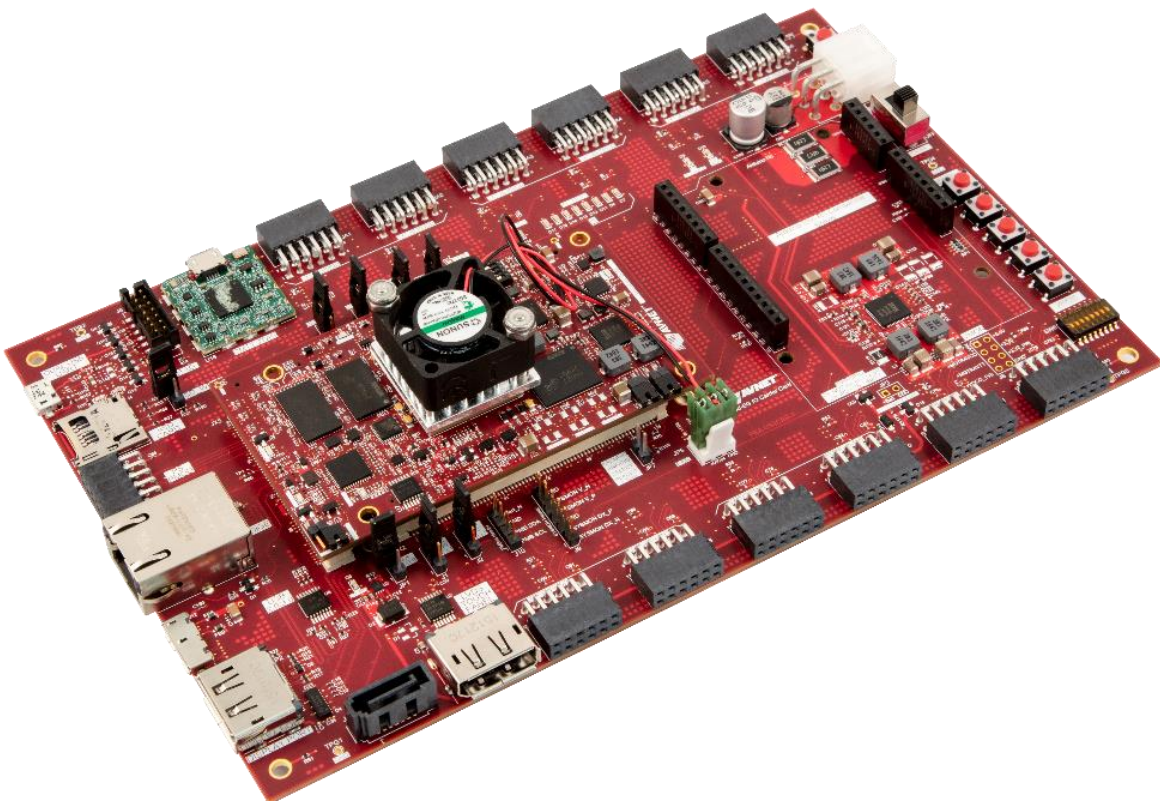
This document describes a Zynq UltraScale+ embedded system design implemented and tested on the Avnet UltraZed-EG Starter Kit and running the PetaLinux operating system with integrated software applications and kernel modules.

The Zynq UltraScale+ hardware development for this example design was performed with Xilinx PetaLinux and Vivado v2016.2 tools. Source files for the hardware and software platforms of this embedded system are provided in the Avnet software and HDL git repositories.

UltraZed-EG Starter Kit Overview

The UltraZed-EG™ Starter Kit from Avnet provides engineers with everything needed to develop edge-to-cloud Internet-connected solutions which need to connect to cloud computing services such as IBM Watson. The kit is based on Avnet's UltraZed-EG System-on-Module (SOM) with a Xilinx Zynq® UltraScale+™ MPSoC. Several Pmod™ compatible expansion ports and an Arduino shield interface are also provided allowing for further connection of several low-cost sensor modules.

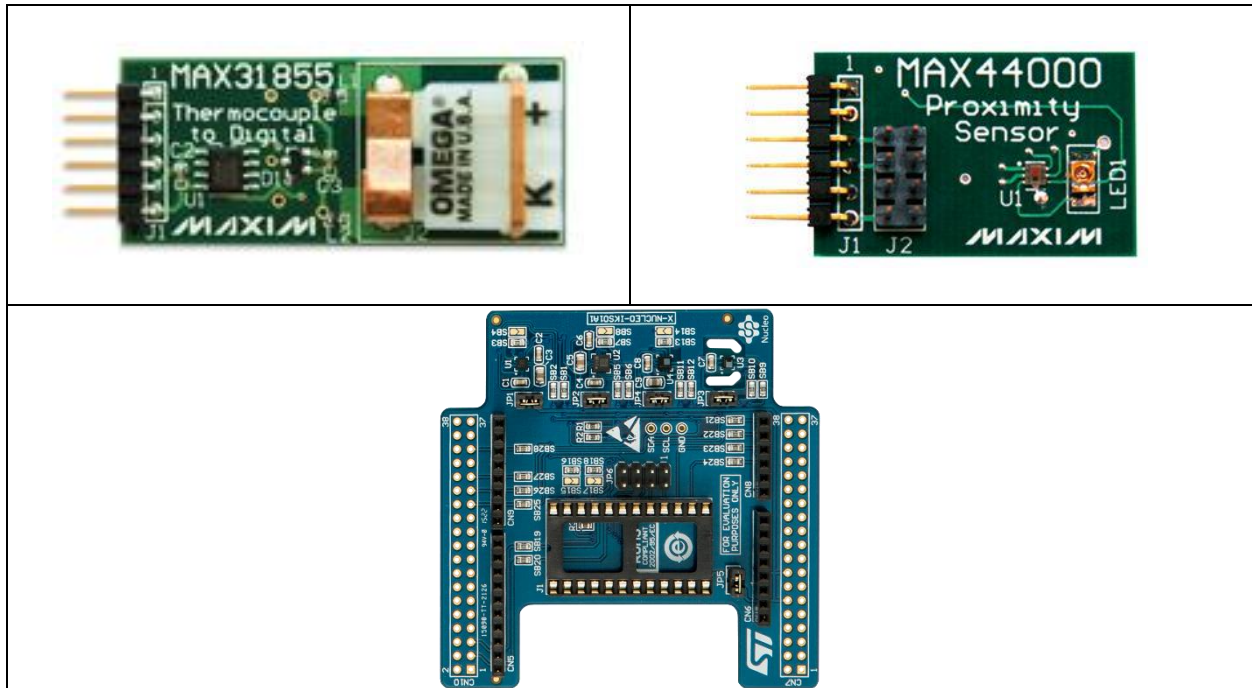
The versatility of this platform offers an excellent prototyping or proof-of-concept vehicle for your new product. Once you are finished prototyping your new product design and are ready to go into production, most of the components found on this platform can be purchased directly from Avnet. Indeed, the UltraZed-EG SOM is a great way to integrate a complete Zynq UltraScale+ solution into your product without worrying about the design complexities of designing your own chip-down system. Please contact your local Avnet FAE for further details.



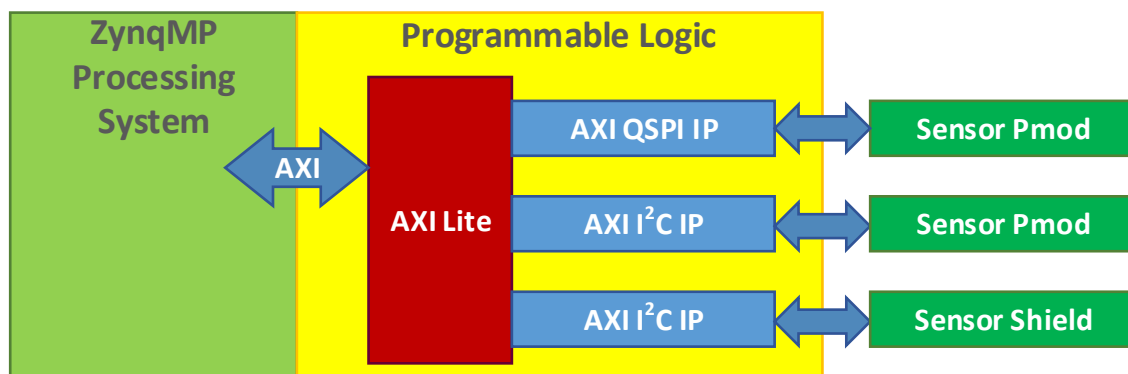
Design Overview

The UltraZed I/O Carrier Card allows the UltraZed-EG System-on-Module (SOM) to connect to a variety of Pmod compatible expansion modules and Arduino “shields”. This example design uses the following sensors:

- Maxim Integrated MAX31855PMB1 thermocouple temperature sensing module.
- Maxim Integrated MAX44000PMB1 proximity sensing module.
- ST Microelectronics MEMS Inertial and Environmental Nucleo Expansion Shield (X-NUCLEO-IKS01A1).



These Pmods can easily be connected to the Programmable Logic of the Xilinx Zynq UltraScale+ MPSoC found on the UltraZed-EG SOM. Since the Processing System (PS) and Programmable Logic (PL) are asynchronous to each other, an AXI interconnect is used to connect the I²C and SPI IP cores to the shield and Pmods. This IP allows the IP cores to capture sensor data from the sensors while the ARM cores are able to perform other important functions.



Objectives

This tutorial is a guide for how to:

- Configure and build the Zynq UltraScale+ MPSoC PS and PL for the UltraZed-EG Starter Kit
- Configure and build the PetaLinux BSP for the UltraZed-EG Starter Kit
- Execute the example design on hardware

Example Design Requirements

Software

The software required to build and execute the example design is:

- Ubuntu Linux 14.4 LTS
- (see http://zedboard.org/sites/default/files/design/VirtualBox_Installation_Guide_2016_2.zip for instructions)
- GtkTerm or another serial terminal emulator
- Git version control manager
- Cloned Avnet HDL and petalinux git repositories
- Xilinx Vivado Design Suite 2016.2
- Xilinx PetaLinux 2016.2

Hardware

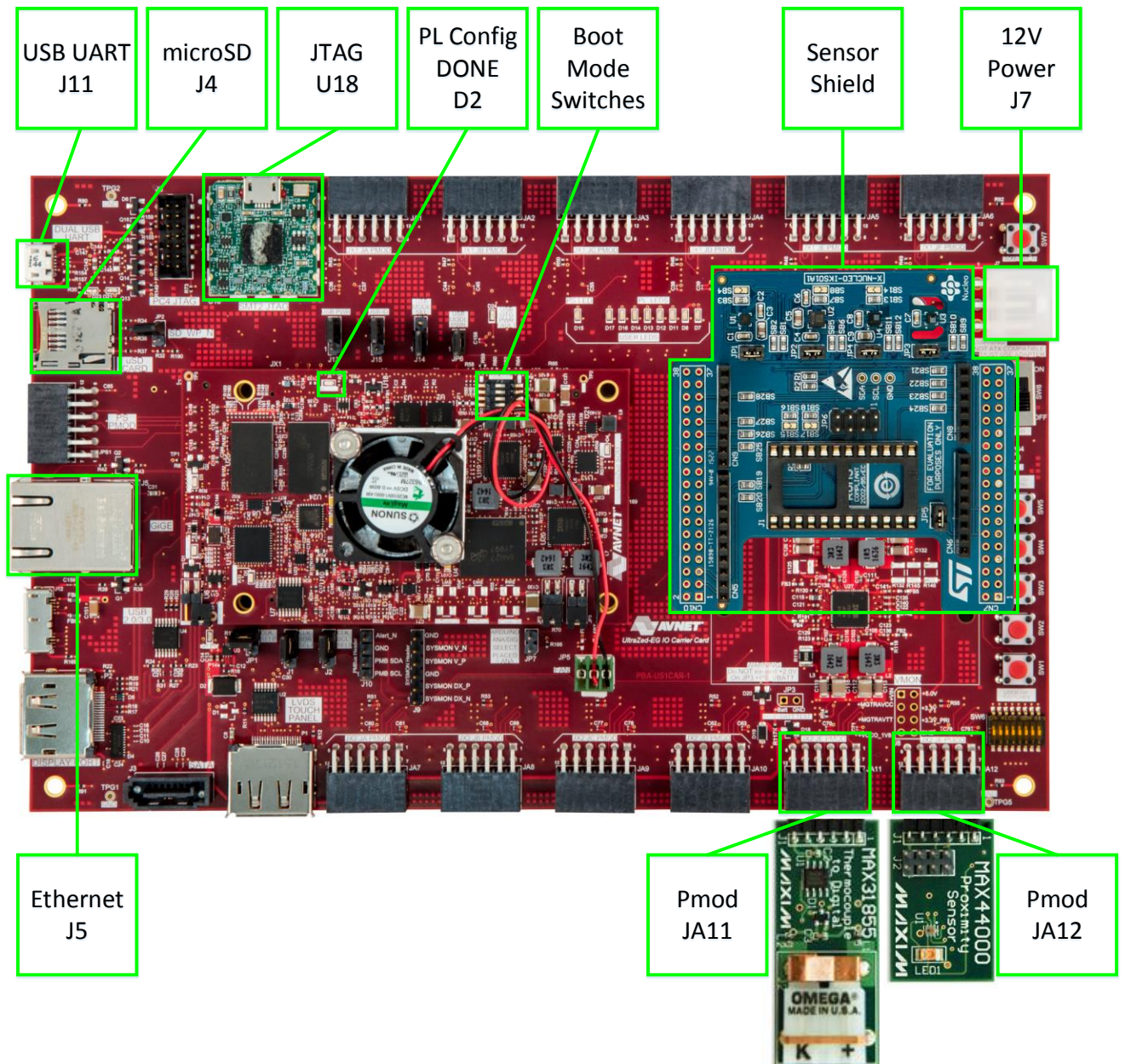
The hardware required to build and execute the reference design is:

- PC with at least 5GB RAM available for Xilinx tools
(<https://www.xilinx.com/products/design-tools/vivado/memory.html>)
 - 4GB required, but 5GB recommended
- Avnet UltraZed-EG Starter Kit (AES-ZU3EGES-1-SK-G)
 - Avnet UltraZed-EG SOM (AES-ZU3EGES-1-SOM-G)
 - Avnet UltraZed I/O Carrier Card (AES-ZU-IOCC-G)
 - 8GB microSD card
 - USB cables
 - Cat-5 Ethernet cable
 - 12V AC/DC supply
- Maxim Integrated Thermocouple Sensor Pmod (MAX31855PMB1)
- Maxim Integrated Proximity Sensor Pmod (MAX44000PMB1)
- ST Microelectronics MEMS Inertial and Environmental Nucleo Expansion Shield (X-NUCLEO-IKS01A1)
 - Arduino R3 Stackable Header Kit (<https://www.sparkfun.com/products/11417>)

Experiment Setup

Setting Up the UltraZed-EG Starter Kit Hardware

Refer to the following figure and perform the following steps to set up the board.



1. Plug the UltraZed-EG SOM onto the IO Carrier Card via JX1/JX2/JX3 connectors and connect the fan to the fan header (JP5) on the IO Carrier Card.
2. Set the UltraZed-EG SOM Boot Mode switch (SW2) (MODE[3:0] = SW2[4:1]) to OFF, OFF, OFF, OFF positions (Boot Mode set to JTAG, MODE[3:0] = 0x0).

3. Install a jumper on the IO Carrier Card JP1.
4. Connect the USB-JTAG port on the I/O Carrier Card (U18) to a free USB port on a PC.
5. Connect the USB-UART port on the I/O Carrier Card (J11) to a free USB port on a PC.
6. Connect the Ethernet port on the I/O Carrier Card (J5) to an Ethernet router/switch or directly to a PC.
7. Plug the Maxim MAX44000 Pmod into the top row of the JA12 Pmod connector.
8. Plug the Maxim MAX31855 Pmod into the top row of the JA11 Pmod connector. Plug the thermocouple probe wire into the Pmod.
9. Plug the STMicroelectronics X-NUCLEO-IKS01A1 Shield and the extension headers into the CON1-4 Arduino connectors on the I/O Carrier Card.
10. Connect the 12V power cable, but do not turn on the board yet.

PC Setup

Install the Zynq UltraScale+ ES1 License

Use the software voucher shipped with the UltraZed-EG Starter Kit to obtain and install the Vivado license to unlock the Zynq UltraScale+ ES1 devices.

Install the Vivado Board Definition Files

A set of Vivado Board Definition Files are provided for the UltraZed-EG Starter Kit in order to automate the hardware platform generation. Please unzip the following file:

<installation>\Vivado_files\AES-ZU3EGES-1-SOM-G-Board_Definition_Files_v2016_2_Release.zip

To the following folder of the Vivado 2016.2 install directory:

<Xilinx_install>\Vivado\2016.2\data\boards\board_files

Enable the Zynq UltraScale+ ES1 Devices in Vivado

- Create a text file called init.tcl and enter the following line:
enable_beta_device*
- Save and close the init.tcl file.
- Place the init.tcl file in the **<Xilinx_install>\Vivado\2016.2\scripts** folder

Setup the Linux Development Host

This example design requires the Xilinx Vivado, SDK, and PetaLinux tools be installed and licensed on a 64-bit Linux host PC. This can be a 64-bit Linux virtual machine running on a 64-bit Windows host, or a native install. Instructions for creating and configuring the Linux virtual machine and installing the Xilinx tools can be found in the [VirtualBox and Linux VM Installation Guide v2016.2](#) document.

Set the Xilinx Vivado and PetaLinux Environment Variables

The Xilinx Vivado and PetaLinux tools all require environment variables to be set before they can be used. Follow the instructions below to set these correctly.

Notes

- Xilinx recommends using the bash command shell for PetaLinux development. Please set your default shell as bash or else there may be issues when building PetaLinux.
- You must run the settings script each time you open a new terminal window or shell. The PetaLinux SDK and Xilinx tools will not operate correctly otherwise.
- You must source the Xilinx Vivado **settings64.sh** script *before* the PetaLinux settings script.

1. Set the Vivado environment variables.

- a. Navigate to the folder where the Xilinx Vivado tools are installed:

```
$ cd /<xilinx_install_path>/Xilinx/Vivado/2016.2
```

- b. Execute the shell script to set the required environment variables:

```
$ source settings64.sh
```

- c. Verify they are set correctly

```
$ echo $XILINX_VIVADO
```

You should see the '/<xilinx_install_path>/Xilinx/Vivado/2016.2' path echoed to your command window.

2. Set the PetaLinux environment variables.

- a. Open a command terminal window and navigate to the folder where the PetaLinux SDK is installed:

```
$ cd /<petalinux_install_path>/petalinux-v2016.2-final
```

- b. Execute the shell script to set the required environment variables:

```
$ source settings.sh
```

- c. Verify they are set correctly:

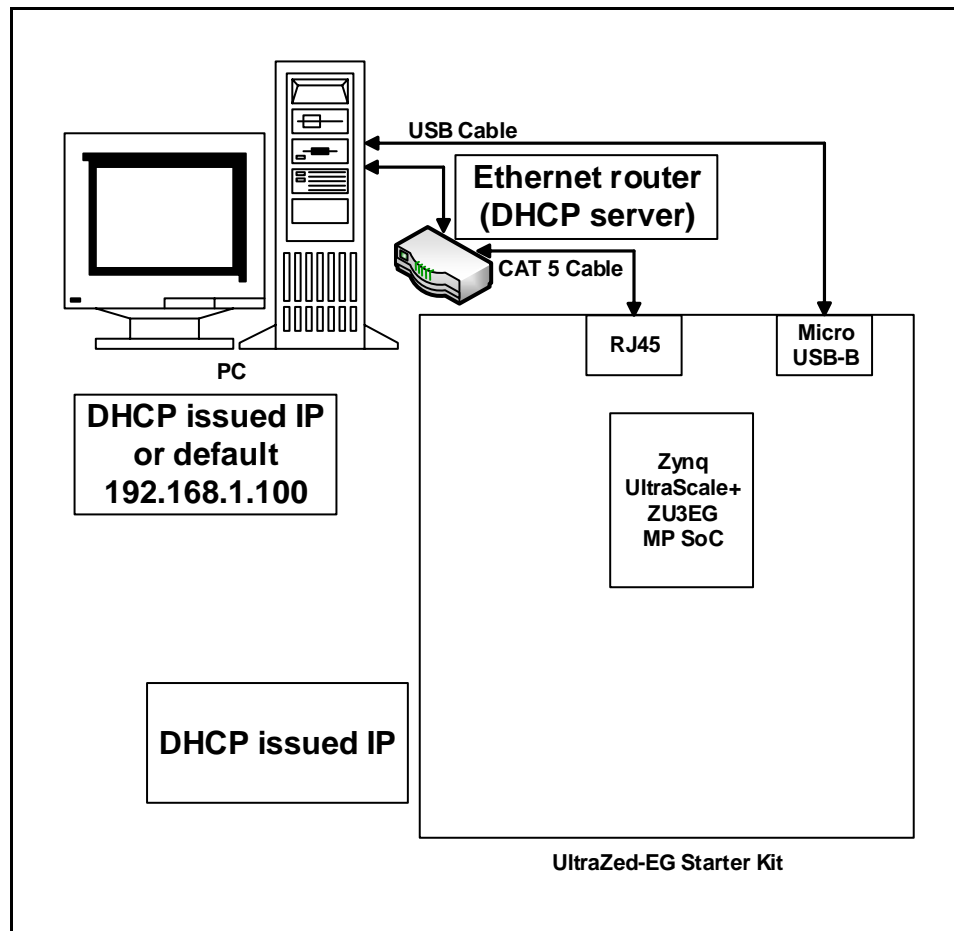
```
$ echo $PETALINUX
```

You should see the '/<petalinux_install_path>/petalinux-v2016.2-final' path echoed to your command window.

3. Do not close this command shell. It will be used again later.

Setup the Ethernet Connection

Follow the steps below to configure the Ethernet NIC of the development host PC to establish an Ethernet connection with the UltraZed-EG Starter Kit.



1. If you choose to connect the UltraZed-EG Starter Kit directly to the host PC and are not running a DHCP server you will need to manually configure the IP address of the host PC and development board. The suggested IP addresses in the table below.

	Host PC	UltraZed-EG Starter Kit
Suggested IP address	192.168.1.100	192.168.1.50
Subnet mask	255.255.255.0	255.255.255.0

2. If you choose to connect the UltraZed-EG Starter Kit directly to the host PC and your PC has a 1000 Mbps capable NIC, you may have to manually set the line rate for the Ethernet interface on the host PC. Gigabit Ethernet does not always auto-negotiate with a direct connection between the development board and host PC.

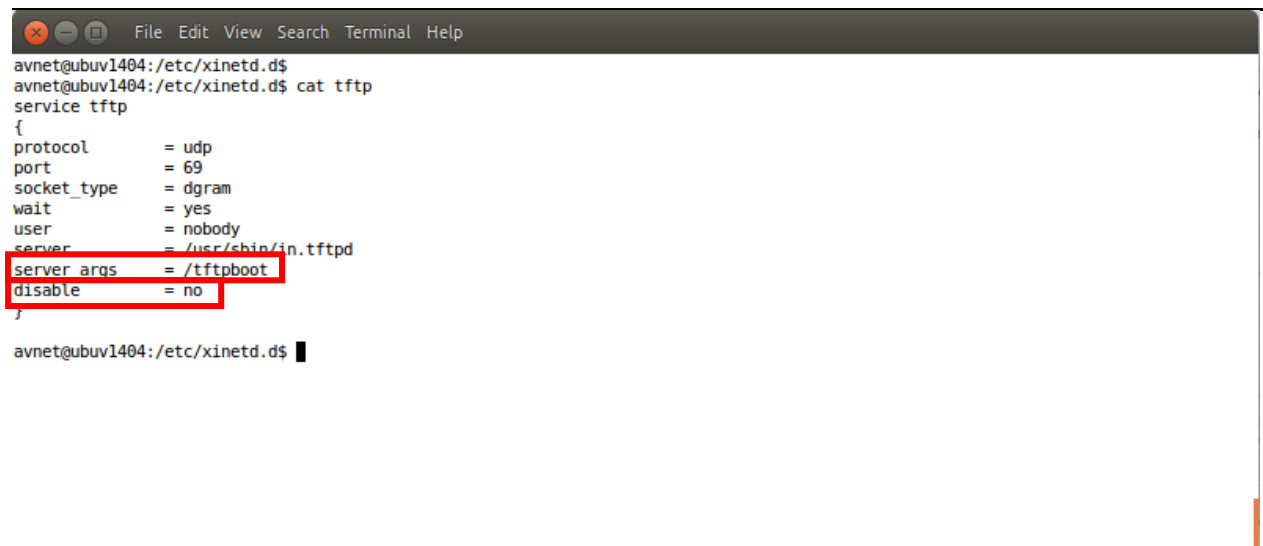
Setup the TFTP Server

The u-boot bootloader requires a TFTP server to be running on the Linux development host in order to download the PetaLinux kernel image to the UltraZed SOM target board for testing.

1. Open a command window.
2. Follow the instructions in the [PetaLinux Installation Guide](#) to install the TFTP server for the Linux distro running on your development host (Ubuntu/Debian, CentOS/RHEL, etc.). You will find this in the section '**PetaLinux Tools Installation Requirements**'.
3. Create a directory in the root of your Linux host where the PetaLinux build files will be copied. By default this is '**/tftpboot**'. You will need to be 'root' user to create this directory. Also be sure to give this directory full access permissions for all users:

```
# chmod 777 /tftpboot
```

4. By default the TFTP server is disabled. To enable the TFTP server, edit /etc/xinetd.d/tftp file as the 'root' user, replacing the word '**yes**' on the 'disable' line with the word '**no**'. Also verify the '/tftpboot' directory is specified on the 'server_args' line. Save the file and exit the editor.



```
avnet@ubuv1404:/etc/xinetd.d$  
avnet@ubuv1404:/etc/xinetd.d$ cat tftp  
service tftp  
{  
protocol      = udp  
port          = 69  
socket_type   = dgram  
wait          = yes  
user          = nobody  
server        = /usr/sbin/in.tftpd  
server_args   = /tftpboot  
disable       = no  
}
```

5. If your system is running a desktop Linux distribution such as Red Hat or CentOS Linux, which starts and stops system processes by using run configuration (rc) scripts, you can simply restart the daemon by invoking these scripts in one of the following commands that is appropriate for your daemon:

```
# /etc/init.d/xinetd restart
# /etc/init.d/inetd restart
```

6. This command will stop and then restart all of the services managed by the daemon on your Linux system. In addition to the **restart** command, you can also issue **stop** and **start** commands this way.

Note: If your Linux system is running Internet services on which other systems depend, restarting the daemon will cause a slight interruption in those services.

7. After executing this command, the TFTP server will be started on your system in response to incoming TFTP requests, and you can access any files you copied to /tftpboot.

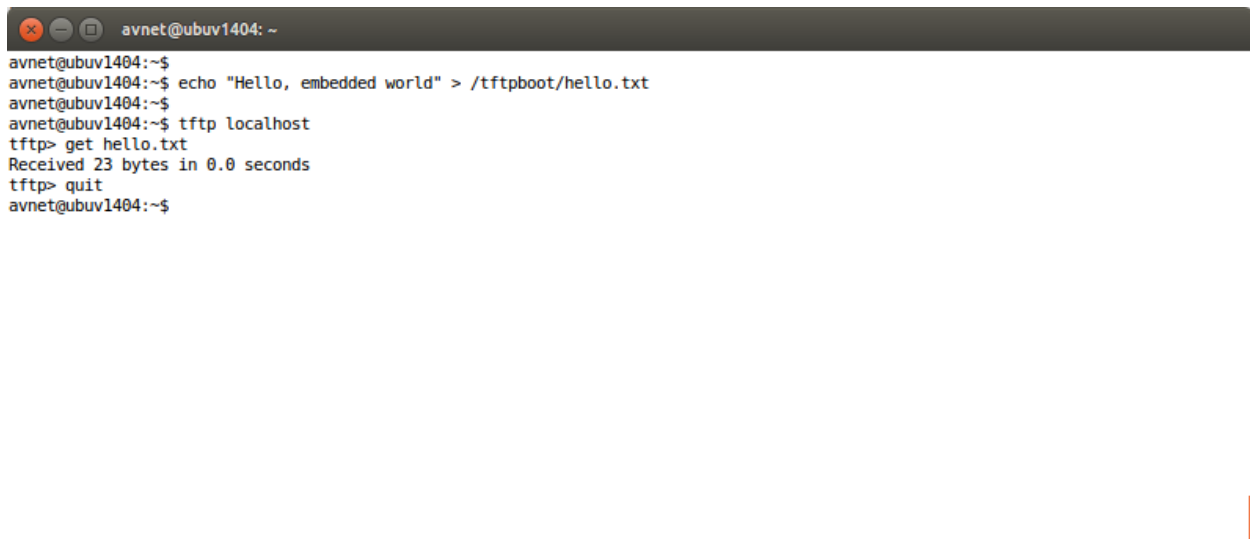
8. Test the tftp server to make sure it is running:

- a. Place a small text file in /tftpboot:

```
$ echo "Hello, embedded world" > /tftpboot/hello.txt
```

- b. Then execute the following commands:

```
$ tftp localhost
tftp> get hello.txt
tftp> quit
```

A terminal window titled 'avnet@ubuv1404: ~' showing the following commands and output:

```
avnet@ubuv1404:~$
avnet@ubuv1404:~$ echo "Hello, embedded world" > /tftpboot/hello.txt
avnet@ubuv1404:~$
avnet@ubuv1404:~$ tftp localhost
tftp> get hello.txt
Received 23 bytes in 0.0 seconds
tftp> quit
avnet@ubuv1404:~$
```

Supplied Files

The following directory structure is included with this reference design:

doc: Contains the documentation for this reference design.

UZ3EG_IOCC_Sensors_PetaLinux_PL2016_2.pdf: This document.

vivado_files: Contains the files to add to the Vivado installation.

AES-ZU3EGES-1-SOM-G-Board_Definition_Files_v2016_2_Release.zip: Board definition files.

init.tcl: TCL script to enable Zynq UltraScale+ device targets with ES1 silicon.

Reusable Components

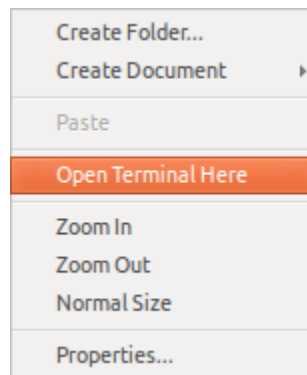
The tutorial will make use of components that can be reused in your own designs:

- Bash shell script: automatically build Vivado hardware design and PetaLinux BSP
 - **make_uz_iocc_sensor.sh:** Builds entire project for UltraZed-EG.
- TCL script: automatically build Vivado design from source code
 - **make_uz_iocc_sensor.tcl:** Builds entire project for UltraZed-EG.
- XDC constraints: Defines pinout and constraints for various carriers
 - **uz_iocc_sensor.xdc:** Constraints for UltraZed-EG Starter Kit platform

Experiment 1: Clone the Avnet HDL Repository

In this section, the design files for the reference design will be retrieved from the Avnet hdl git repository.

1. Extract the design archive of this tutorial to a folder where you have full read and write permissions and at least 12GB of free space. The path to this folder should not have any spaces in it. This folder is referred to later in this tutorial as the <installation> folder.
2. Using your favorite file explorer, navigate to the folder where the design archive was extracted and open a command window.



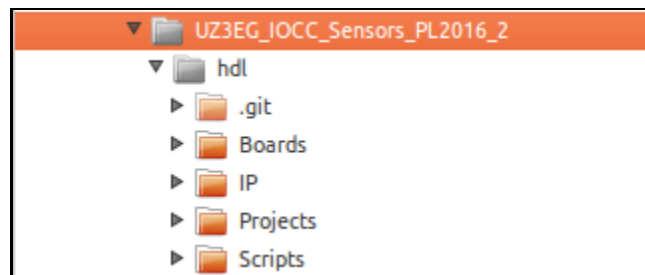
3. Clone the Avnet/hdl git repository.

```
$ git clone https://github.com/Avnet/hdl.git
```

4. Navigate to the hdl repository folder and checkout the tag for the hardware platform designed to work with this tutorial into a new branch.

```
$ cd hdl
$ git checkout -b devel_branch \
uz_iocc_sensor_UZ3EG_IOCC_20170511_115000
```

5. You should see the following directory structure:



6. The **<installation>/hdl** repository contains the following sub-directories:

Directory	Content Description
<installation>/hdl/ Boards	contains board related files
<installation>/hdl/ IP	contains the IP cores used by the ref designs
<installation>/hdl/ Projects	contains project related files
<installation>/hdl/ Scripts	contains scripts used to automatically build the designs

For the UltraZed-EG Sensors example design, the following content is of interest:

Directory	Content Description
<installation>/hdl/Projects/ uz_iocc_sensor	Folder containing files for reference design on UltraZed-EG Starter Kit
<installation>/hdl/Scripts/ make_uz_iocc_sensor.tcl	TCL script to launch the build of the reference design for a UltraZed-EG Starter target

Experiment 2: Build the UltraZed-EG Sensors Vivado Project

In this section, the design files for the reference design will be retrieved from the Avnet software git repository.

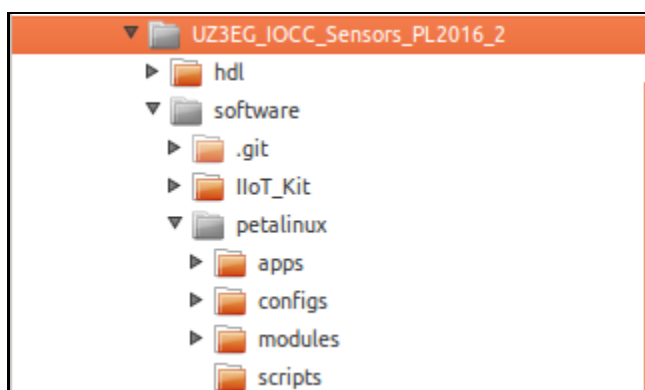
1. Navigate back to the <installation> folder and clone the Avnet/software git repository.

```
$ cd ..  
$ git clone https://github.com/Avnet/software.git
```

2. Navigate to the software repository folder and checkout the tag for the software platform designed to work with this tutorial into a new branch.

```
$ cd software  
$ git checkout -b devel_branch \  
uz_iocc_sensor_UZ3EG_IOCC_20170511_115000
```

3. You should see the following directory structure:



4. The <installation>/software repository contains the following sub-directories:

Directory	Content Description
<installation>/software/IIoT_Kit	contains board related files
<installation>/software/petalinux/apps	contains the linux apps used in various PetaLinux BSPs
<installation>/software/petalinux/configs	contains PetaLinux project related config files
<installation>/software/petalinux/modules	contains kernel modules used in various PetaLinux BSPs
<installation>/software/petalinux/scripts	contains scripts used to automatically build the BSPs

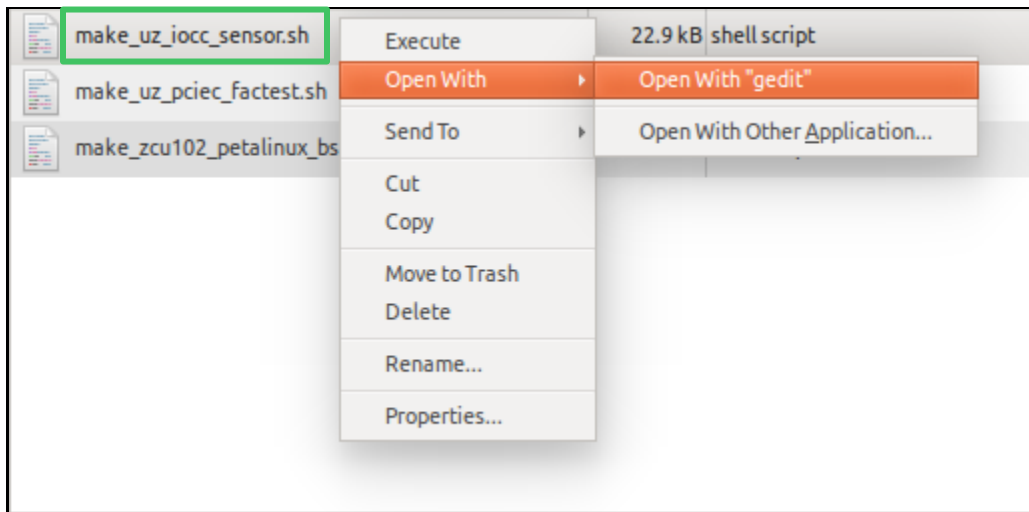
For the UltraZed-EG PetaLinux Sensors example design, the following content is of interest:

Directory	Content Description
<installation>/software/scripts/make_uz_iocc_sensor.sh	Bash shell script to launch the build of the hardware platform and PetaLinux BSP for a UltraZed-EG Starter Kit target

Experiment 3: Build the Vivado Design and PetaLinux BSP

In this section we examine the build script for this tutorial and use it to build the Zynq UltraScale+ design in Vivado and export it for use in the PetaLinux system.

1. Navigate to the <installation>/software/petalinux/scripts folder and use your favorite text editor to open the `make_uz_iocc_sensor.sh` shell script that will build the Vivado hardware platform and PetaLinux BSP for this tutorial.



2. This shell script fully automates the building of the Vivado hardware platform and the PetaLinux BSP. These are tasks that are a bit complicated, time consuming, and difficult to document in a tutorial like this. Still, though, it is very helpful to understand what this script is doing. There are many commands that are run in this shell script, but some of the key commands are described below:
 - a) Scroll to the end of the file and examine the **main_make_function()** set of commands. This function is responsible for first creating the hardware platform needed for generating the PetaLinux BSP. The following command starts the Vivado tools in batch mode and runs the commands in the named TCL file to build the Zynq UltraScale+ MPSoC design.

```
vivado -mode batch -source make_${HDL_PROJECT_NAME}.tcl
```

```
504 # Change to HDL scripts folder.
505 cd ${START_FOLDER}/${HDL_SCRIPTS_FOLDER}
506
507 # Launch vivado in batch mode to build hardware platforms for target
508 # boards.
509 vivado -mode batch -source make_${HDL_PROJECT_NAME}.tcl
510
511 #
512 # Create the PetaLinux BSP for the UZ3EG_IOCC target.
513 #
514 HDL_BOARD_NAME=UZ3EG_IOCC
515 PETALINUX_PROJECT_NAME=uz3eg_iocc_sensor_2016_2
516 create_petalinux_bsp
517
```

- b) The next steps are part of the **create_petalinux_bsp()** function. Once the Zynq UltraScale+ MPSoC hardware design has been built in Vivado the next key steps are:
- Create the PetaLinux project for the zynqMP target
petalinux-create --type project --template zynqMP --name \${PETALINUX_PROJECT_NAME}
 - Import the hardware description
petalinux-config --oldconfig --get-hw-description=./hw_platform/ -p \${START_FOLDER}/\${PETALINUX_PROJECTS_FOLDER}/\${PETALINUX_PROJECT_NAME}

```

180
181 # Create the PetaLinux project.
182 petalinux-create --type project --template zynqMP --name ${PETALINUX_PROJECT_NAME}
183
184 # Create the hardware definition folder.
185 mkdir -p ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}/hw_platform
186
187 # Import the hardware definition files and hardware platform bitstream from
188 # implemented system products folder.
189 cd ${START_FOLDER}/${HDL_PROJECTS_FOLDER}
190
191 echo " "
192 echo "Importing hardware definition ${HDL_HARDWARE_NAME} from impl_1 folder ..."
193 echo " "
194
195 cp -f ${HDL_PROJECT_NAME}/${HDL_BOARD_NAME}/${HDL_PROJECT_NAME}.runs/impl_1/${HDL_PROJECT_NAME}_wrapper.sysdef \
196 ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}/hw_platform/${HDL_HARDWARE_NAME}.hdf
197
198 echo " "
199 echo "Importing hardware bitstream ${HDL_HARDWARE_NAME} from impl_1 folder ..."
200 echo " "
201
202 cp -f ${HDL_PROJECT_NAME}/${HDL_BOARD_NAME}/${HDL_PROJECT_NAME}.runs/impl_1/${HDL_PROJECT_NAME}_wrapper.bit \
203 ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}/hw_platform/system_wrapper.bit
204
205 # Change directories to the hardware definition folder for the PetaLinux
206 # project, at this point the .hdf file must be located in this folder
207 # for the petalinux-config step to be successful.
208 cd ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}
209
210 # Import the hardware description into the PetaLinux project.
211 petalinux-config --oldconfig --get-hw-description=./hw_platform/ -p ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}
212

```

- c) One of the big time savers this script does is that it automates the adding of the software applications, shell scripts, and kernel modules that we want to add to the root file system of the PetaLinux system. Here is one example:
- Create the application and enable it in the PetaLinux project
petalinux-create --type apps --name demo_setup --enable
 - Copy the source files from the repository folder into the PetaLinux project
**cp -rf \${START_FOLDER}/\${PETALINUX_APPS_FOLDER}/demo_setup/* **
\${START_FOLDER}/\${PETALINUX_PROJECTS_FOLDER}/\${PETALINUX_PROJECT_NAME}/
components/apps/demo_setup

```

228 # Create a PetaLinux application named demo_setup.
229 petalinux-create --type apps --name demo_setup --enable
230
231 # Copy the demo_setup application information over to the demo_setup
232 # application folder.
233 cp -rf ${START_FOLDER}/${PETALINUX_APPS_FOLDER}/demo_setup/* \
234 ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}/components/apps/demo_setup
235
236 # Create a PetaLinux application named generic_buffer.
237 petalinux-create --type apps --name generic_buffer --enable
238
239 # Copy the generic_buffer application information over to the generic_buffer
240 # application folder.
241 cp -rf ${START_FOLDER}/${PETALINUX_APPS_FOLDER}/generic_buffer/* \
242 ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}/components/apps/generic_buffer
243

```

- d) Copy the known-good configuration files for the PetaLinux rootfs, devicetree, and kernel into the PetaLinux project. For example, the rootfs config file:

```
cp -rf ${START_FOLDER}/${PETALINUX_CONFIGS_FOLDER}/rootfs/config.UZ3EG_IOCC_SENSOR \
${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}/
subsystems/linux/configs/rootfs/config
```

```
380 # Overwrite the rootfs component config with the revision controlled source
381 # config.
382 echo " "
383 echo "Overwriting rootfs config ..."
384 echo " "
385 cp -rf ${START_FOLDER}/${PETALINUX_CONFIGS_FOLDER}/rootfs/config.UZ3EG_IOCC_SENSOR \
386 ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}/subsystems/linux/configs/rootfs/config
387
388 # Overwrite the top level devicetree source with the revision controlled
389 # source file.
390 echo " "
391 echo "Overwriting top level devicetree source ..."
392 echo " "
393 cp -rf ${START_FOLDER}/${PETALINUX_CONFIGS_FOLDER}/device-tree/system-top.dts.UZ3EG_IOCC_SENSOR \
394 ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}/subsystems/linux/configs/device-tree/system-top.dts
395
396 # Overwrite the config level devicetree source with the revision controlled
397 # source file.
398 echo " "
399 echo "Overwriting config level devicetree source ..."
400 echo " "
401 cp -rf ${START_FOLDER}/${PETALINUX_CONFIGS_FOLDER}/device-tree/system-conf.dtsi.UZ3EG_IOCC_SENSOR \
402 ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}/subsystems/linux/configs/device-tree/system-conf.dtsi
403
404 # Overwrite the kernel component config with the revision controlled source
405 # config.
406 echo " "
407 echo "Overwriting kernel config ..."
408 echo " "
409 cp -rf ${START_FOLDER}/${PETALINUX_CONFIGS_FOLDER}/kernel/config.UZ3EG_IOCC_SENSOR \
410 ${START_FOLDER}/${PETALINUX_PROJECTS_FOLDER}/${PETALINUX_PROJECT_NAME}/subsystems/linux/configs/kernel/config
411
```

- e) Then, finally, clean and build the PetaLinux project. It is always a good idea to do a clean before building the PetaLinux system. This insures the new build is starting with a clean slate and there aren't any outdated files that might otherwise cause problems during the build process.

```
petalinux-build -x distclean
petalinux-build
```

```
422
423 # Make sure that intermediary files get cleaned up.
424 petalinux-build -x distclean
425
426 # Build PetaLinux project.
427 petalinux-build
428
```

- ```
$ cd <installation>/software/petalinux/scripts
$./make_uz_iocc_sensor.sh
```

Page 19

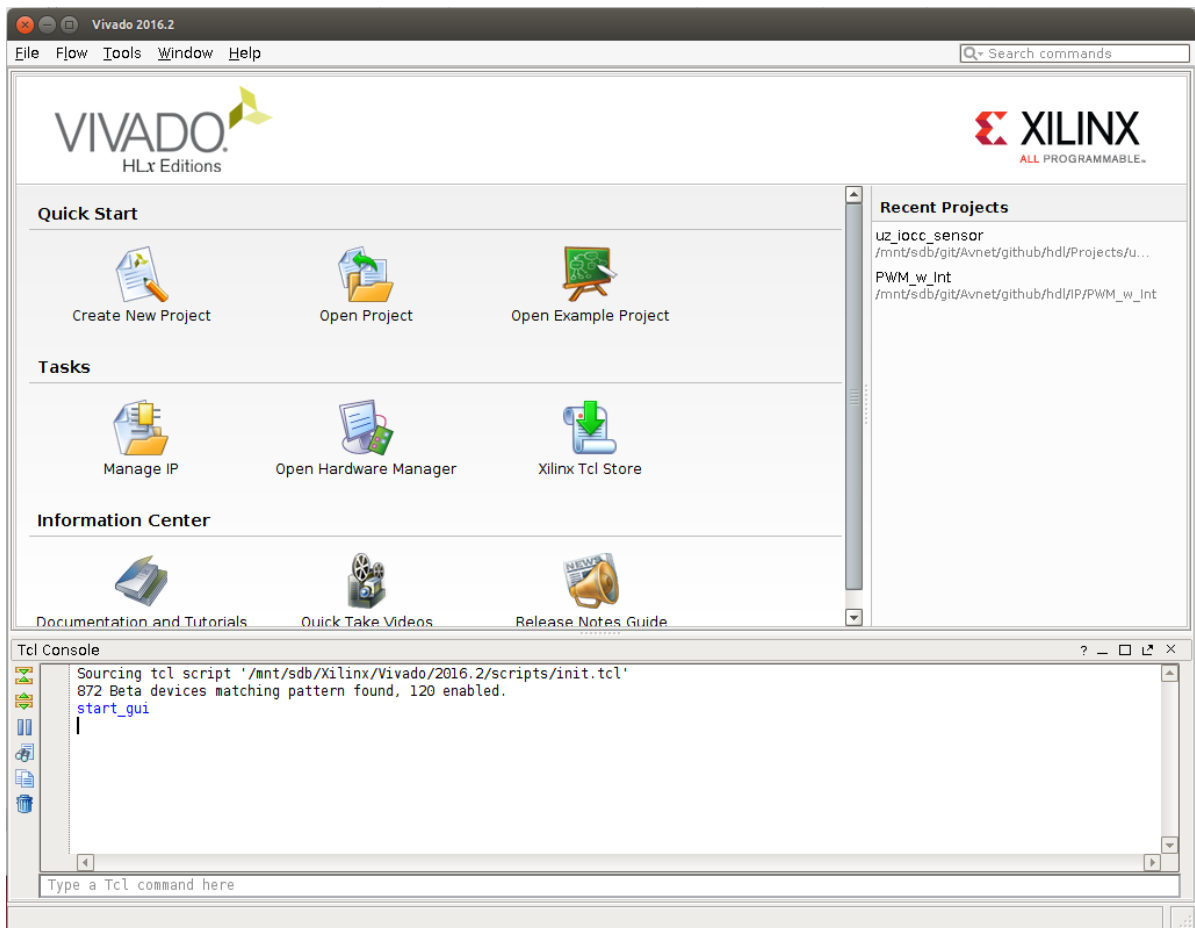
## Experiment 4: Examine the UltraZed Hardware Platform

It is important to understand the Zynq UltraScale+ MPSoC PS and PL hardware platform the Linux OS and software applications, etc. will be running on.

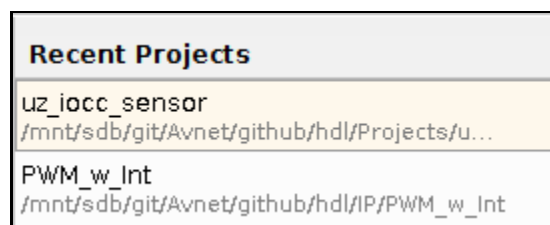


1. Launch Vivado either by double-clicking the desktop icon, or from the command window.

**\$ vivado**



2. Open the completed design project by clicking on the **uz\_iocc\_sensor** project in the **Recent Projects** list.

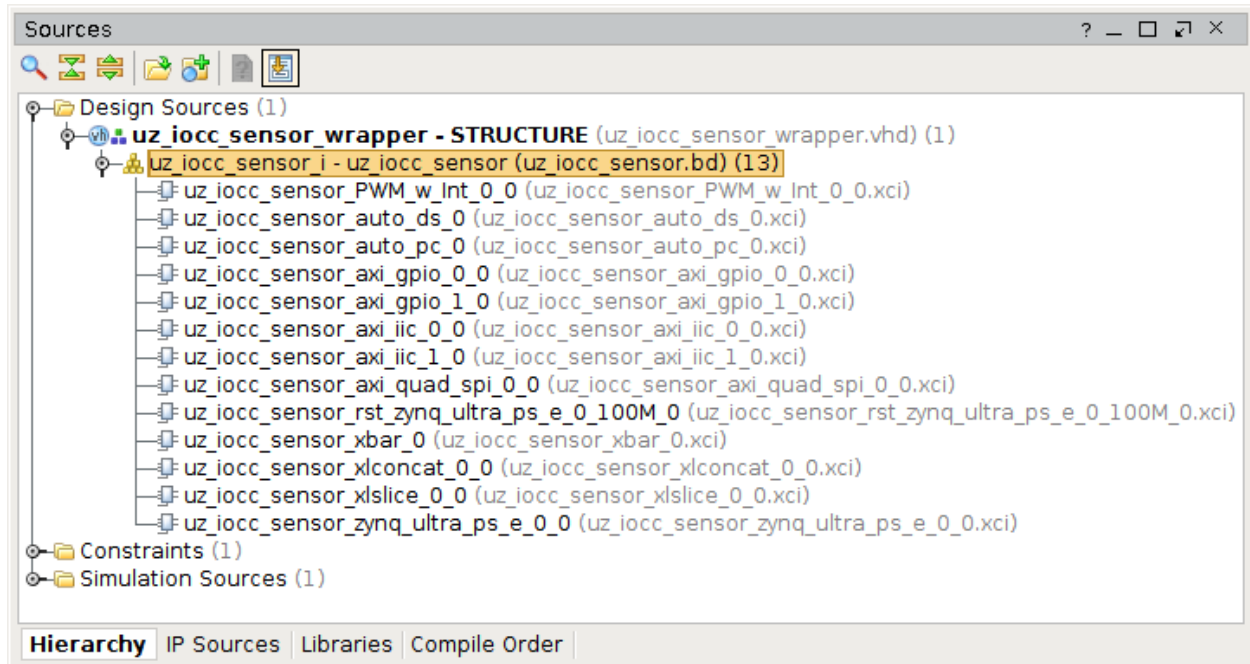


3. In the **Project Summary** tab, you can find many high level details about the design such as an Implemented Timing Report, Device Utilization Report, and Power Report.

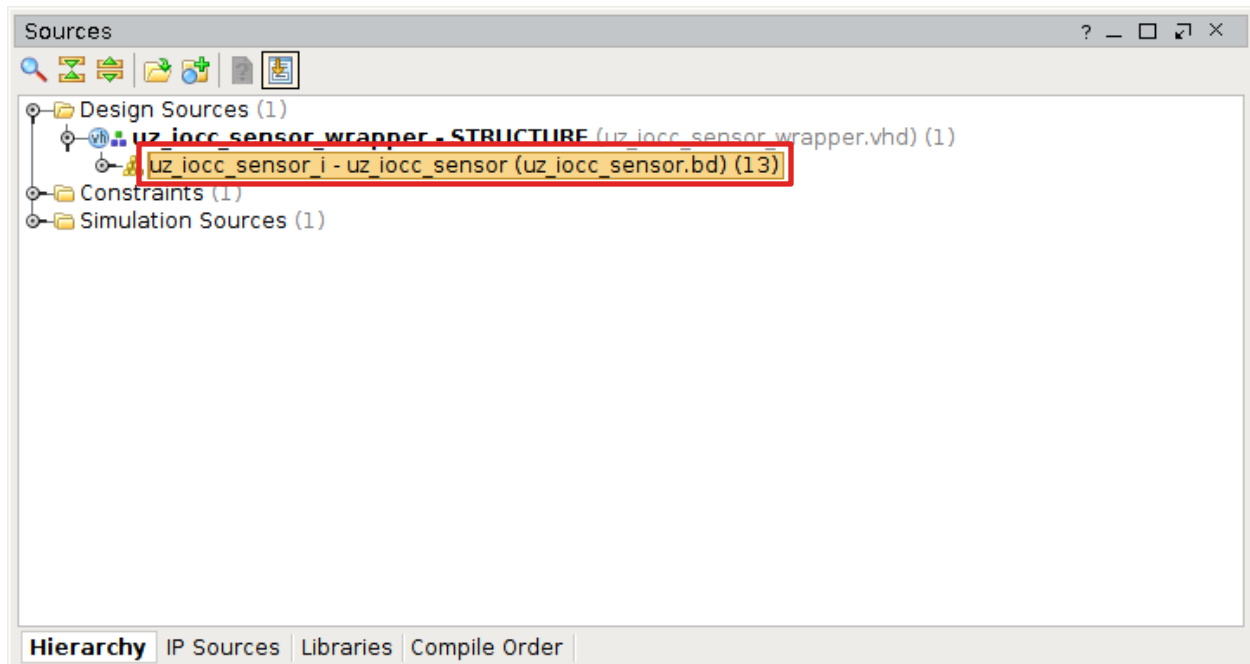
The screenshot shows the 'Project Summary' window in the Vivado IDE. The window has a title bar with standard window controls. The main content is organized into several sections:


- Project Settings**: Includes fields for Project name (uz\_iocc\_sensor), Project location (/mnt/sdb/git/Avnet/github/hdl/Projects/uz\_iocc\_sensor/UZ3EG\_I0CC), Product family (Zynq UltraScale+), Project part (Avnet UltraZed-3EG IO Carrier Card (xczu3eg-sfva625-1-i-es1)), Top module name (uz\_iocc\_sensor\_wrapper), Target language (VHDL), and Simulator language (Mixed). An 'Edit' link is in the top right.
- Board Part**: Includes Display name (Avnet UltraZed-3EG IO Carrier Card), Board part name (em.avnet.com:ultrazed\_eg\_iocc:part0:1.0), Repository path (/mnt/sdb/Xilinx/Vivado/2016.2/data/boards/board\_files), URL (http://www.ultrazed.org), and Board overview (Avnet UltraZed-3EG IO Carrier Card).
- Synthesis**: Status (Not started), Messages (No errors or warnings), Part (xczu3eg-sfva625-1-i-es1), and Strategy (Vivado Synthesis Defaults).
- Implementation**: Status (Not started), Messages (No errors or warnings), Part (xczu3eg-sfva625-1-i-es1), Strategy (Vivado Implementation Defaults), and Incremental compile (None).
- DRC Violations**: A link to 'Run Implementation' to see DRC results.
- Timing**: A link to 'Run Implementation' to see timing results.
- Utilization**: A link to 'Run Synthesis' to see utilization results.
- Power**: A link to 'Run Implementation' to see power results.

4. In the Sources tab, expand the **uz\_iocc\_sensor\_wrapper** branch of the **Design Sources** tree to see the items under the top level design wrapper.



5. Double-click the **uz\_iocc\_sensor.bd** entry to open the block design for the project

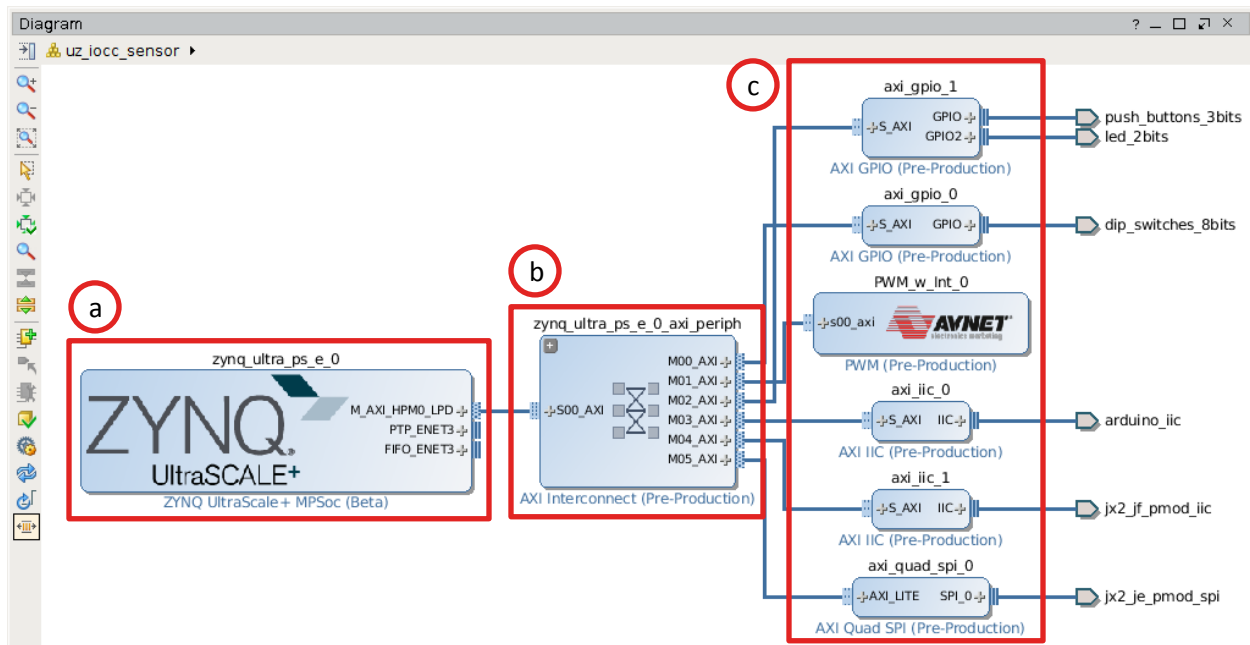


6. Explore the block design **Diagram** tab that is now opened. Click on the  icon to limit the window to show only the interface connections. This will make the diagram easier to read. Note that there are three major components for this basic design:

- a. Zynq UltraScale+ Processing System (zynq\_ultra\_ps\_e\_0)
- b. AXI Interconnect which acts as means to move data between the processor and the peripherals in the PL (zynq\_ultra\_ps\_e\_0\_axi\_periph)
- c. AXI peripherals for GPIO, I2C, QSPI, and PWM

Vivado uses this block design to drive the synthesis and implementation capability of this tool to generate a Programmable Logic bitstream file which can be loaded into the FPGA portion of the Zynq UltraScale+ device.

This block design is very customizable and could easily be used as a basis for expansion with other sensor modules to fit a particular application.



7. Please close Vivado before continuing.

## Experiment 5: Boot PetaLinux Using TFTP

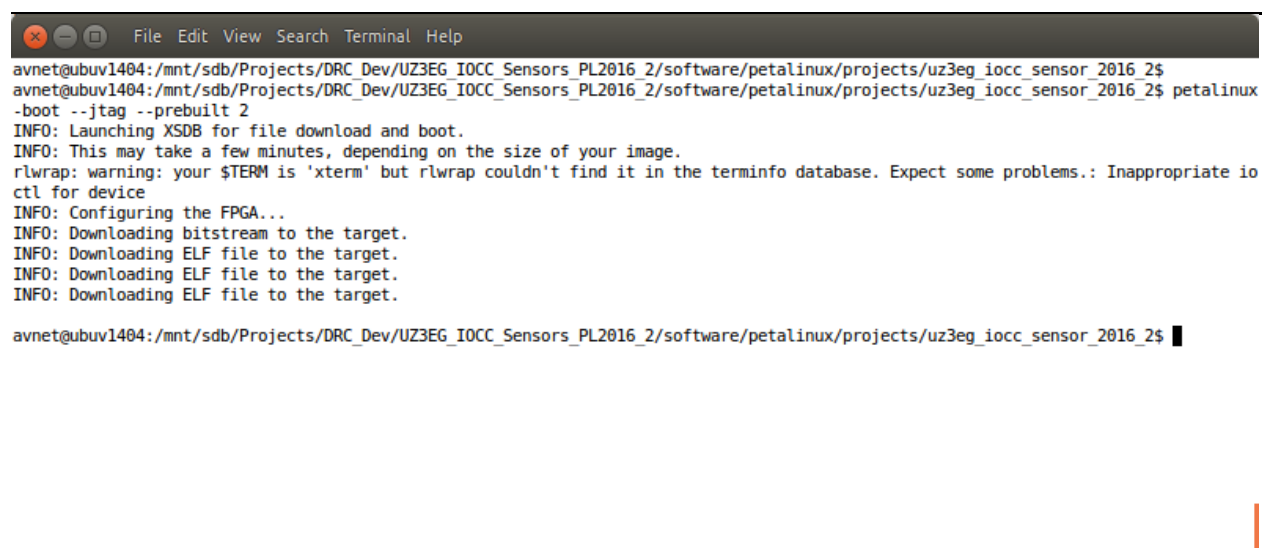
A very useful method to verify that the UltraZed SOM will boot our new PetaLinux system correctly is to use JTAG to configure the Zynq UltraScale+ PL, download and execute the FSBL and u-boot bootloader, and fetch the kernel image and root file system via the Trivial File Transfer Protocol (TFTP). This method is especially convenient if your u-boot build is stable, but you are still working on your PetaLinux kernel development or root file system. This way, changes to the kernel or root file system can be tested and deployed quickly. Follow the steps below to use the **petalinux-boot** command to boot PetaLinux on the UltraZed-EG using TFTP to fetch the kernel and root file system image.

1. If not already done, connect the UltraZed-EG Starter Kit to the host PC as described earlier in [Setting Up the UltraZed-EG Starter Kit Hardware](#). Turn on the UltraZed-EG Starter Kit. The power switch (SW8) is near the 12V power connector on the I/O Carrier Card.
2. If not already done, follow the instructions to [Setup the TFTP Server](#).
3. Start a serial terminal session and set the serial port parameters to **tttyUSB1, 115200** baud rate, **no** parity, **8** bits, **1** stop bit and no flow control.
4. Go to the command window we opened earlier and change directories to the PetaLinux project folder:

```
$ cd <installation>/software/petalinux/projects/uz3eg_iocc_sensor_2016_2
```

5. Type the following command to configure the Zynq UltraScale+ PL then download and execute the u-boot bootloader and PetaLinux kernel:

```
$ petalinux-boot --jtag --prebuilt 2
```



```
avnet@ubuv1404:/mnt/sdb/Projects/DRC_Dev/UZ3EG_I0CC_Sensors_PL2016_2/software/petalinux/projects/uz3eg_iocc_sensor_2016_2$
avnet@ubuv1404:/mnt/sdb/Projects/DRC_Dev/UZ3EG_I0CC_Sensors_PL2016_2/software/petalinux/projects/uz3eg_iocc_sensor_2016_2$ petalinux
-boot --jtag --prebuilt 2
INFO: Launching XSD for file download and boot.
INFO: This may take a few minutes, depending on the size of your image.
rlwrap: warning: your $TERM is 'xterm' but rlwrap couldn't find it in the terminfo database. Expect some problems.: Inappropriate io
ctl for device
INFO: Configuring the FPGA...
INFO: Downloading bitstream to the target.
INFO: Downloading ELF file to the target.
INFO: Downloading ELF file to the target.
INFO: Downloading ELF file to the target.
avnet@ubuv1404:/mnt/sdb/Projects/DRC_Dev/UZ3EG_I0CC_Sensors_PL2016_2/software/petalinux/projects/uz3eg_iocc_sensor_2016_2$
```

- When u-boot starts there is a 2 second delay to allow the user to stop u-boot from automatically booting the Linux kernel. Since we are only booting u-boot and not the Linux kernel and root file system you may let the autoboot timer expire. When u-boot has finished booting you should see the following in your serial terminal window.

```
GtkTerm - /dev/ttyUSB1 115200-8-N-1

U-Boot 2016.01 (May 17 2017 - 23:48:54 -0400)

DRAM: 2 GiB
Enabling Caches...
EL Level: EL2
MMC: sdhci@ff160000: 0, sdhci@ff170000: 1
SF: Detected N25Q256A with page size 512 Bytes, erase size 8 KiB, total 64 MiB
In: serial
Out: serial
Err: serial
Bootmode: JTAG_MODE
Net: ZYNQ GEM: ff0e0000, phyaddr 5, interface rgmii-id
eth0: ethernet@ff0e0000
Hit any key to stop autoboot: 0
Device: sdhci@ff160000
Manufacturer ID: 13
OEM: 14e
Name: Q2J55
Tran Speed: 52000000
Rd Block Len: 512
MMC version 5.0
High Capacity: Yes
Capacity: 7.1 GiB
Bus Width: 8-bit
Erase Group Size: 512 KiB
HC WP Group Size: 8 MiB
User Capacity: 7.1 GiB WRREL
Boot Capacity: 16 MiB ENH
RPMB Capacity: 4 MiB ENH
sdhci_send_command: MMC: 1 busy timeout increasing to: 200 ms.
sdhci_send_command: MMC: 1 busy timeout increasing to: 400 ms.
sdhci_send_command: MMC: 1 busy timeout increasing to: 800 ms.
sdhci_send_command: MMC: 1 busy timeout increasing to: 1600 ms.
sdhci_send_command: MMC: 1 busy timeout increasing to: 3200 ms.
sdhci_send_command: MMC: 1 busy timeout.
sdhci_send_command: MMC: 1 busy timeout.
** Bad device mmc 1 **
U-Boot-PetaLinux>
```

/dev/ttyUSB1 115200-8-N-1 DTR RTS CTS CD DSR RI

7. The first time U-Boot is run you might see a console message that says **\*\*\* Warning - bad CRC, using default environment.** Type the following command at the U-Boot prompt to initialize the area of flash where the U-Boot environment variables are stored to stop this warning message:  
**U-Boot-PetaLinux> saveenv**

```
GtkTerm - /dev/ttyUSB1 115200-8-N-1
Release 2016.2 May 17 2017 - 23:47:46
Platform: Silicon, Cluster ID 0x80000000
Running on A53-0 (64-bit) Processor
Processor Initialization Done
===== In Stage 2 =====
In JTAG Boot Mode
===== In Stage 4 =====
Exit from FSBL
NOTICE: ATF running on XCZU3EG/silicon v2/RTL5.1 at 0xfffe5000
NOTICE: BL31: Secure code at 0x60000000
NOTICE: BL31: Non secure code at 0x80000000
NOTICE: BL31: v1.2(release):8825d70
NOTICE: BL31: Built : 23:47:50, May 17 2017

U-Boot 2016.01 (May 17 2017 - 23:48:54 -0400)

DRAM: 2 GiB
Enabling Caches...
EL Level: EL2
MMC: sdhci@ff160000: 0, sdhci@ff170000: 1
SF: Detected N25Q256A with page size 512 Bytes, erase size 8 KiB, total 64 MiB
*** Warning - bad CRC, using default environment

Error: flags type check failure for "serverip" <= "AUTO" (type: i)
himport_r: can't insert "serverip=AUTO" into hash table
In: serial
Out: serial
Err: serial
Bootmode: JTAG_MODE
Net: ZYNQ GEM: ff0e0000, phyaddr 5, interface rgmii-id
eth0: ethernet@ff0e0000
U-BOOT for uz3eg-2016-2

Hit any key to stop autoboot: 0
U-Boot-PetaLinux> saveenv
Saving Environment to SPI Flash...
SF: Detected N25Q256A with page size 512 Bytes, erase size 8 KiB, total 64 MiB
Erasing SPI flash...Writing to SPI flash...done
U-Boot-PetaLinux>
```

/dev/ttyUSB1 115200-8-N-1 DTR RTS CTS CD DSR RI

8. The u-boot bootloader will fetch the PetaLinux image over Ethernet from the Linux host using TFTP, so we need to configure u-boot with the IP address of the Linux host that is running the TFTP server. Run the **ifconfig** command on the Linux host to identify its IP address.

```
avnet@ubuv1404:/mnt/sdb/Projects/DRC_Dev/UZ3EG_IOC_Sensors_PL2016_2/software/petalinux/projects/uz3eg_iocc_sensor_2016_2$ ifconfig
eth0 Link encap:Ethernet HWaddr 08:00:27:d2:28:69
 inet addr:192.168.1.156 Bcast:192.168.1.255 Mask:255.255.255.0
 inet6 addr: fe80::a00:27ff:fed2:2869/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:67428 errors:0 dropped:0 overruns:0 frame:0
 TX packets:49996 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:4614126 (4.6 MB) TX bytes:27744642 (27.7 MB)

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 inet6 addr: ::1/128 Scope:Host
 UP LOOPBACK RUNNING MTU:65536 Metric:1
 RX packets:19949 errors:0 dropped:0 overruns:0 frame:0
 TX packets:19949 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1
 RX bytes:141373897 (141.3 MB) TX bytes:141373897 (141.3 MB)

avnet@ubuv1404:/mnt/sdb/Projects/DRC_Dev/UZ3EG_IOC_Sensors_PL2016_2/software/petalinux/projects/uz3eg_iocc_sensor_2016_2$
```

9. Set the u-boot **serverip** environment variable with the host's IP address. If the IP address has been previously set and saved using **saveenv** and hasn't changed, then it won't be necessary to set it again now. Otherwise, set the **serverip** now:

```
U-Boot-PetaLinux> printenv serverip
U-Boot-PetaLinux> setenv serverip <host_IP_address>
```

10. Likewise, u-boot also has to be configured with its own IP address. If the UltraZed-EG Start Kit is connected to a LAN with a DHCP server you can use the u-boot dhcp command to fetch an IP address.

```
U-Boot-PetaLinux> dhcp
```

```
U-Boot-PetaLinux> dhcp
BOOTP broadcast 1
BOOTP broadcast 2
DHCP client bound to address 192.168.1.136 (1232 ms)
U-Boot-PetaLinux>
```

Alternatively if there isn't a DHCP server, or if the UltraZed-EG Starter Kit is connected via Ethernet directly to the Linux host PC, then use the setenv command to manually set an IP address. Make sure to specify an address that is currently unused on your LAN.

```
U-Boot-PetaLinux> setenv ipaddr <target_IP_address>
```

- There is a u-boot command macro named **netboot** that will issue the tftp command to fetch the kernel image(**image.ub**), place it at a set location in DDR4 memory(**\${netstart}**), and issue the boot command (**bootm**) to begin booting the kernel image:  
**netboot=tftpboot \${netstart} {kernel img} && bootm**

```
U-Boot-PetaLinux> run netboot
```

You will notice hashtags scroll in the serial console indicating that the kernel is being fetched from the tftp server and placed in DDR4 memory.

```
GtkTerm - /dev/ttyUSB2 115200-8-N-1
Rd Block Len: 512
MMC version 5.0
High Capacity: Yes
Capacity: 7.1 GiB
Bus Width: 8-bit
Erase Group Size: 512 KiB
HC WP Group Size: 8 MiB
User Capacity: 7.1 GiB WRREL
Boot Capacity: 16 MiB ENH
RPMB Capacity: 4 MiB ENH
sdhci_send_command: MMC: 1 busy timeout increasing to: 200 ms.
sdhci_send_command: MMC: 1 busy timeout increasing to: 400 ms.
sdhci_send_command: MMC: 1 busy timeout increasing to: 800 ms.
sdhci_send_command: MMC: 1 busy timeout increasing to: 1600 ms.
sdhci_send_command: MMC: 1 busy timeout increasing to: 3200 ms.
sdhci_send_command: MMC: 1 busy timeout.
sdhci_send_command: MMC: 1 busy timeout.
** Bad device mmc 1 **
U-Boot-PetaLinux> setenv serverip 192.168.1.141
U-Boot-PetaLinux> dhcp
BOOTP broadcast 1
BOOTP broadcast 2
DHCP client bound to address 192.168.1.142 (1236 ms)
U-Boot-PetaLinux> run netboot
Using ethernet@ff0e0000 device
TFTP from server 192.168.1.141: our IP address is 192.168.1.142
Filename 'image.ub'.
Load address: 0x10000000
Loading: #####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
#####
```

12. Once the kernel boots you will recognize the familiar login screen:

```
GtkTerm - /dev/ttyUSB2 115200-8-N-1
/etc/udhcpd.d/50default: Adding DNS 192.168.1.1
done.
Starting Dropbear SSH server: Generating key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCFc+ckXa1PCyZmy7ZCcn9gKmj08unE5rJYPkuHyhW0ReandstU6KGh+z2FZ0HY81+ukeTj+Jodz1t7mUOU6VPv
FxfkopA08RCKIypVe03kmQRfqHI6kYTzEGjbRwSZupdnwRvFkM89/rMRLine8yoUM/R0/N1RzoSrx552frzZ2+g06tN1kNF6Hxtg5ocTFR0+CNFvEw/SwmpyVIg
k1qjUQuB3AQiWZ85mflVqCIAvQooLDlt[9.931499] hts221 driver: init
H3vRw2278+uav7Dvg/laxd2geErbCSTXHW2KCJc1ugKtpBMv7ekStjLUfUiqBKv7[9.937797] Inside hts221_probe
3CERchTEEyai4Te6JtxBYuph4VU+Agag8D7X root@uz3eg-iocc-sensor-2016[9.946402] hts221 1-005f: probe start.
-2
Fingerprint: md5 f1:01:3e:94:07:f8:64:2b:de:b1:d3:d3:26:eb:[9.955868] hts221 1-005f: using default plaform_data for humi
dity
15:88
dropbear.
starting Busybox HTTP Daemon: httpd... done.[9.967528] hts221: hw init start

*****[9.976797] hts221: hw init done

*** Avnet UltraZed SOM + I/O CC IoT Sensor Demo Launcher ***

[10.232220] input: hts221 as /devices/platform/amba_pl/80030000.i2c/i2c-1/1-005f/input/input0
[10.245877] hts221 1-005f: hts221: probed
[12.254546] lps25hb: probe start.
[12.358173] lps25hb 1-005d: using default plaform_data for lps25
[12.426165] lps25hb ID Chip OK
[12.490166] lps25hb: hw init start
[12.498039] input: lps25hb as /devices/platform/amba_pl/80030000.i2c/i2c-1/1-005d/input/input1
[12.507039] lps25hb 1-005d: lps25hb: probed
[14.516394] lis3mdl_mag driver: init
[14.519963] lis3mdl_mag 1-001e: probe start.
[14.524254] lis3mdl_mag: hw init start
[14.529058] lis3mdl_mag: hw init done
[14.534443] input: lis3mdl_mag as /devices/platform/amba_pl/80030000.i2c/i2c-1/1-001e/input/input2
[14.543768] lis3mdl_mag 1-001e: lis3mdl_mag: probed

uz3eg-iocc-sensor-2016-2 login: █

/dev/ttyUSB2 115200-8-N-1 DTR RTS CTS CD DSR RI
```

13. Login to the kernel with the following credentials:

User: **root**

Password: **root**

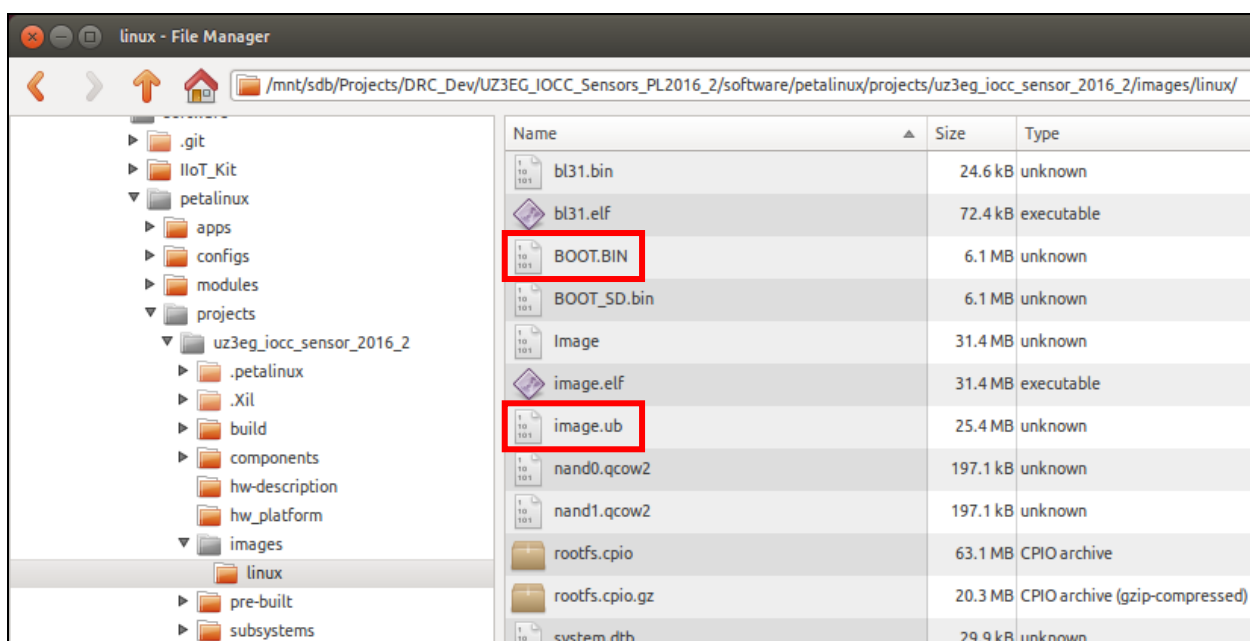
## Experiment 6: Boot PetaLinux from the microSD Card

The image creation process is now complete. All that remains is to transfer the image files to the boot media and power up the board to see PetaLinux in action.

The PetaLinux kernel image, device tree, and root file system are contained in a single file named **image.ub**. This file was created in the kernel build step in [Experiment 3](#).

The boot image (**BOOT.BIN**) created by the PetaLinux packager contains the Zynq UltraScale+ FSBL, the PL bitstream and u-boot executable.

1. Use your favorite file manager and navigate to the `<installation>/software/petalinux/projects/uz3eg_iocc_sensor_2016_2/images/linux/` folder. Copy the **BOOT.BIN** and the **image.ub** files to the top level directory of a FAT or FAT32 formatted microSD card.



2. Make the cable connections to your target board as described in [Setting Up the UltraZed-EG Starter Kit Hardware](#). If you previously booted PetaLinux using JTAG as described [Boot PetaLinux Using TFTP](#) you will need to turn off the board, insert the microSD card, and change the boot mode switch settings. Set the UltraZed-EG SOM Boot Mode switch (SW2) (MODE[3:0] = SW2[4:1]) to ON, OFF, ON, OFF positions (SD Card, MODE[3:0] = 0xA).
3. Turn on the UltraZed-EG Starter Kit. The power switch (SW8) is near the 12V power connector on the I/O Carrier Card.

4. Open your serial terminal window to view the PetaLinux boot messages and login prompt.

```
GtkTerm - /dev/ttyUSB1 115200-8-N-1
Lease of 192.168.1.136 obtained, lease time 86400
/etc/udhcpd.d/50default: Adding DNS 192.168.1.1
done.
Starting Dropbear SSH server: Generating key, this may take a while...
Public key portion is:
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQJCJLYbNwoAokb1URIHuznDZ0E4CpC0fmRi2QXaBUx2DC8G0iZqn4u5xAOH0KoeLm+TeQftnNk4g0gponXD0ZCnd
hRHvU78RxiB+rWIJPD+o15r8bCT0cMnQ9RDeod4snBE4YWzsnsgsDTkXb1nivy29XIiv04rvPxCDFPu5/09ap0Ha8qM3d6WcGi+o5B4AQyz5WLU+s+d9TGQJY9/cEi
DBB3fjgvdxmSKm0HpcPabll0FyPD0UVwIpfrnjLX[16.302112] hts221 driver: init
0ZIGDs0JZmixYxu4woMRIGFXSELh06h3xKd/tJNUPLaA1+P0mVc9Jxpjc0iqod89[16.308277] Inside hts221_probe
GwEYe1EVzwVjjlqM0wT76MP+bmrN root@uz3eg-iocc-sensor-2016-2
Fin[16.316896] hts221 1-005f: probe start.
gerprint: md5 ec:66:92:f4:e3:fl:l0:ee:f5:d6:3e:77:64:l0:cf:74
[16.326368] hts221 1-005f: using default plaform_data for humidity
dropbear.
starting Busybox HTTP Daemon: httpd... done.

*[16.337986] hts221: hw init start
*****[16.347283] hts221: hw init done
*
*** ***
*** Avnet UltraZed SOM + I/O CC IoT Sensor Demo Launcher ***
*** ***

[16.604177] input: hts221 as /devices/platform/amba_pl/80030000.i2c/i2c-1/1-005f/input/input0
[16.617804] hts221 1-005f: hts221: probed
[18.626934] lps25hb: probe start.
[18.734130] lps25hb 1-005d: using default plaform_data for lps25
[18.802123] lps25hb ID Chip OK
[18.866122] lps25hb: hw init start
[18.873970] input: lps25hb as /devices/platform/amba_pl/80030000.i2c/i2c-1/1-005d/input/input1
[18.882992] lps25hb 1-005d: lps25hb: probed
[20.891367] lis3mdl_mag driver: init
[20.894943] lis3mdl_mag 1-001e: probe start.
[20.899225] lis3mdl_mag: hw init start
[20.904032] lis3mdl_mag: hw init done
[20.909413] input: lis3mdl_mag as /devices/platform/amba_pl/80030000.i2c/i2c-1/1-001e/input/input2
[20.918762] lis3mdl_mag 1-001e: lis3mdl_mag: probed

uz3eg-iocc-sensor-2016-2 login: █

/dev/ttyUSB1 115200-8-N-1 DTR RTS CTS CD DSR RI
```

5. Login to the kernel with the following credentials.

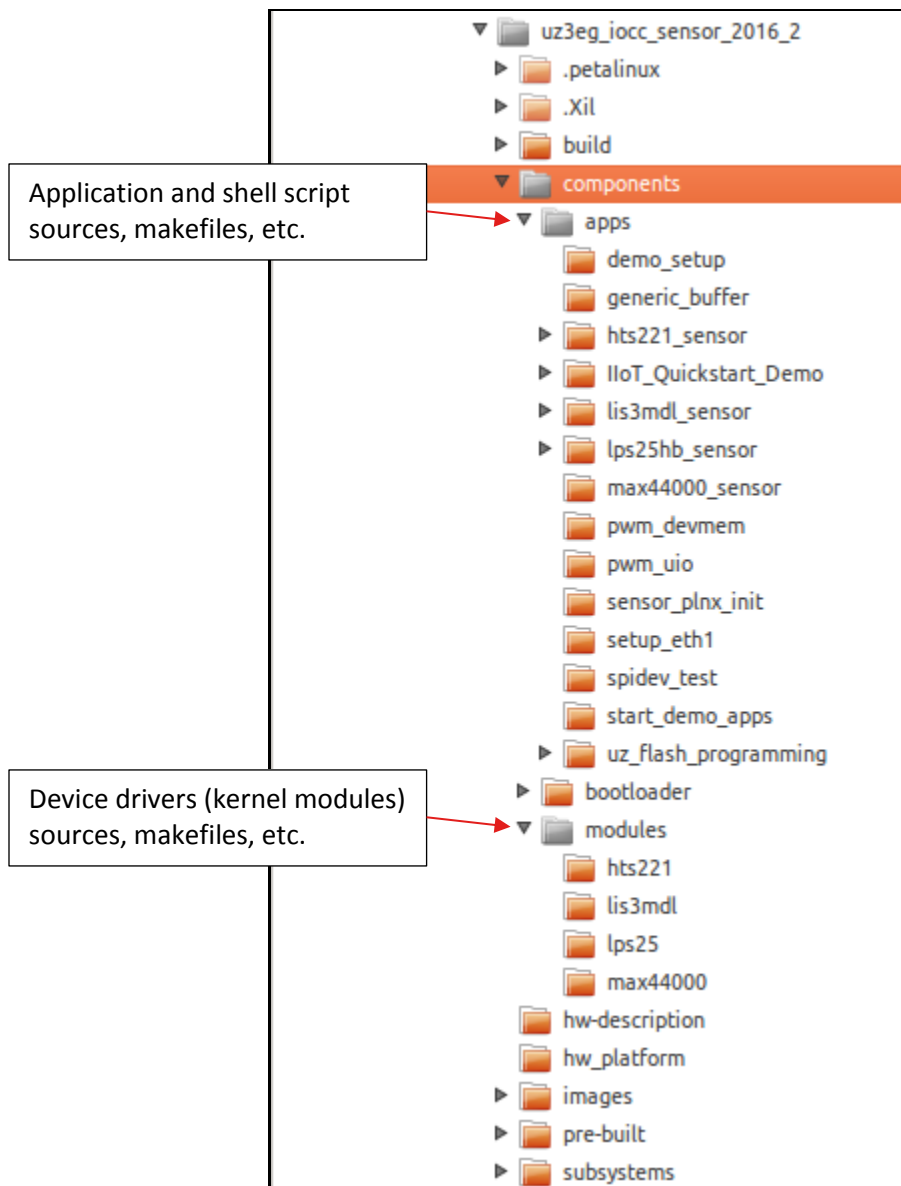
User: **root**  
Password: **root**

## Experiment 7: Run the Supplied Software Applications

Built into the root filesystem are some example software applications to fetch data from the sensors attached to the UltraZed-EG Starter Kit. Some of the applications display the data in the terminal window, and others publish the data to the internet via IBM Bluemix using the MQTT protocol. All of these applications can be used as a starting point for further development

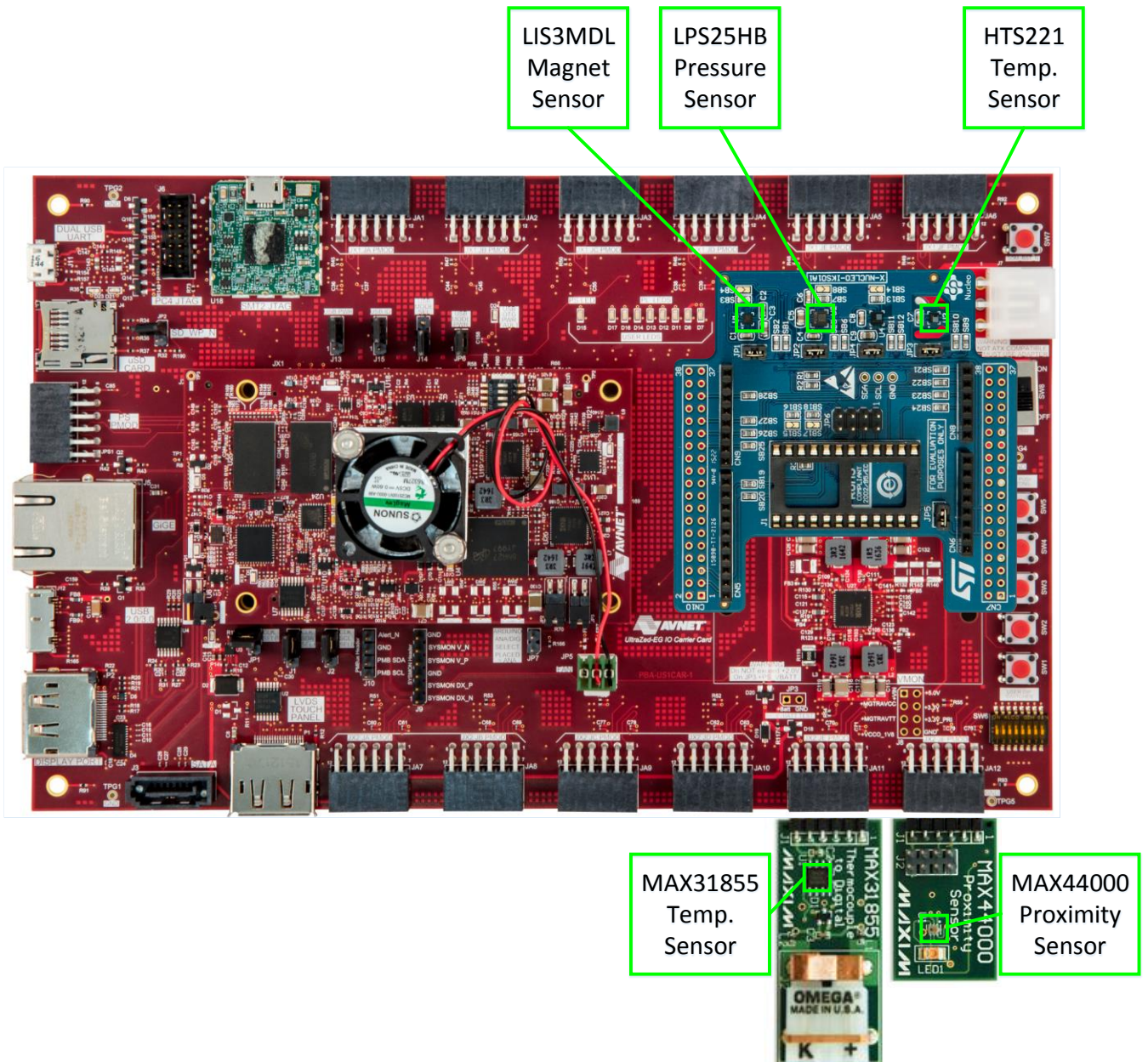
### Location of Software Sources in the PetaLinux Project

The software sources for the sensor device drivers, applications, and shell scripts can be found in the `<installation>/software/petalinux/projects/uz3eg_iocc_sensor_2016_2/components/` folder.



## Location of Sensors

The figure below describes the location of the sensors used with the provided software applications.

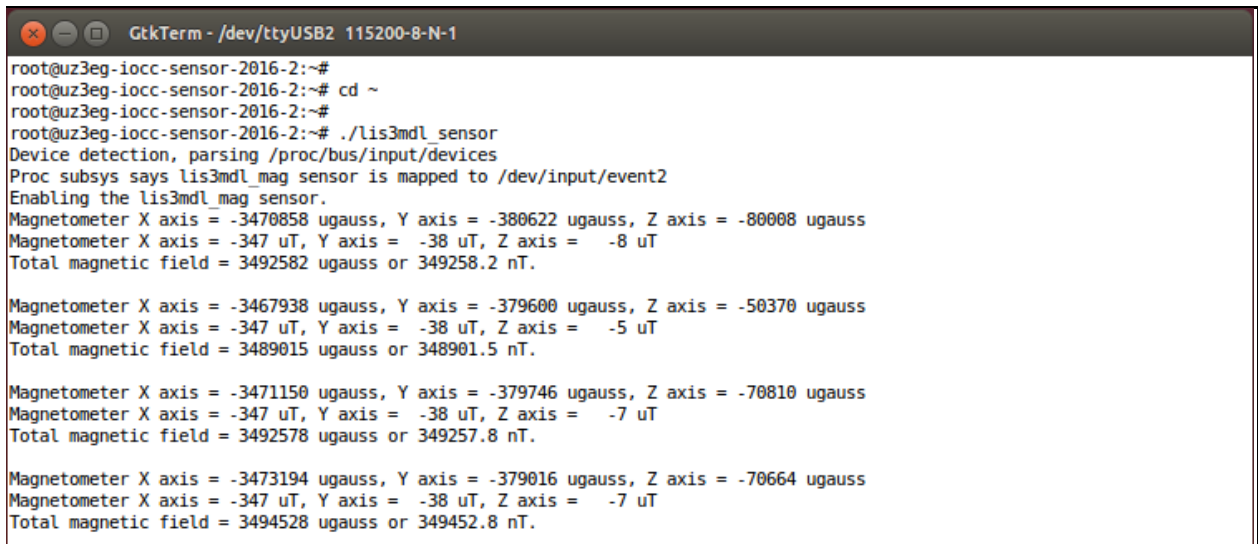


## ST Microelectronics LIS3MDL Magnetometer

The lis3mdl\_sensor software application reads magnetometer data from the ST Microelectronics LIS3MDL sensor and displays it in the terminal window. On the UltraZed-EG Starter Kit, the I<sup>2</sup>C interface is accessed from the application code using the added lis3mdl I<sup>2</sup>C software driver in the PetaLinux kernel.

1. Open the terminal window and navigate to the /home/root folder. Run the application and view the magnetometer data. Pick up the UltraZed-EG Starter Kit assembly (be careful with any attached cables) and tilt it on the x, y, and z axes and watch the data change in the terminal window.

```
cd ~
./lis3mdl_sensor
```



```
GtkTerm - /dev/ttyUSB2 115200-8-N-1
root@uz3eg-iocc-sensor-2016-2:~#
root@uz3eg-iocc-sensor-2016-2:~# cd ~
root@uz3eg-iocc-sensor-2016-2:~#
root@uz3eg-iocc-sensor-2016-2:~# ./lis3mdl_sensor
Device detection, parsing /proc/bus/input/devices
Proc subsys says lis3mdl_mag sensor is mapped to /dev/input/event2
Enabling the lis3mdl_mag sensor.
Magnetometer X axis = -3470858 ugauss, Y axis = -380622 ugauss, Z axis = -80008 ugauss
Magnetometer X axis = -347 uT, Y axis = -38 uT, Z axis = -8 uT
Total magnetic field = 3492582 ugauss or 349258.2 nT.

Magnetometer X axis = -3467938 ugauss, Y axis = -379600 ugauss, Z axis = -50370 ugauss
Magnetometer X axis = -347 uT, Y axis = -38 uT, Z axis = -5 uT
Total magnetic field = 3489015 ugauss or 348901.5 nT.

Magnetometer X axis = -3471150 ugauss, Y axis = -379746 ugauss, Z axis = -70810 ugauss
Magnetometer X axis = -347 uT, Y axis = -38 uT, Z axis = -7 uT
Total magnetic field = 3492578 ugauss or 349257.8 nT.

Magnetometer X axis = -3473194 ugauss, Y axis = -379016 ugauss, Z axis = -70664 ugauss
Magnetometer X axis = -347 uT, Y axis = -38 uT, Z axis = -7 uT
Total magnetic field = 3494528 ugauss or 349452.8 nT.
```

2. Press <CTRL-C> to stop the application.

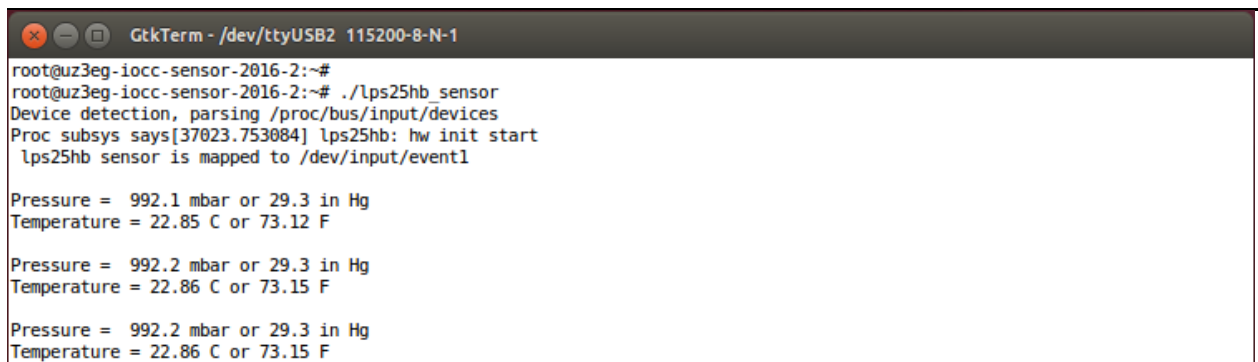
## ST Microelectronics LPS25HB Barometer and Thermometer

The `lps25hb_sensor` software application data from the ST Microelectronics LPS25HB sensor. There are two sensors in the LPS25HB device, one provides absolute pressure and the other provides ambient temperature.

The sample application reads data from both sensors and displays the result on the PC host serial console. On the UltraZed-EG Starter Kit, the I<sup>2</sup>C interface is accessed from the application code using the added `lps25` I<sup>2</sup>C software driver in the PetaLinux kernel.

1. Run the application and view the temperature and pressure data.

```
./lps25hb_sensor
```

A screenshot of a terminal window titled "GtkTerm - /dev/ttyUSB2 115200-8-N-1". The terminal shows the execution of the `lps25hb_sensor` application. The output includes device detection messages and three lines of sensor data, each showing pressure in mbar and inHg, and temperature in Celsius and Fahrenheit.

```
root@uz3eg-iocc-sensor-2016-2:~#
root@uz3eg-iocc-sensor-2016-2:~# ./lps25hb_sensor
Device detection, parsing /proc/bus/input/devices
Proc subsys says[37023.753084] lps25hb: hw init start
lps25hb sensor is mapped to /dev/input/event1

Pressure = 992.1 mbar or 29.3 in Hg
Temperature = 22.85 C or 73.12 F

Pressure = 992.2 mbar or 29.3 in Hg
Temperature = 22.86 C or 73.15 F

Pressure = 992.2 mbar or 29.3 in Hg
Temperature = 22.86 C or 73.15 F
```

2. Press <CTRL-C> to stop the application.

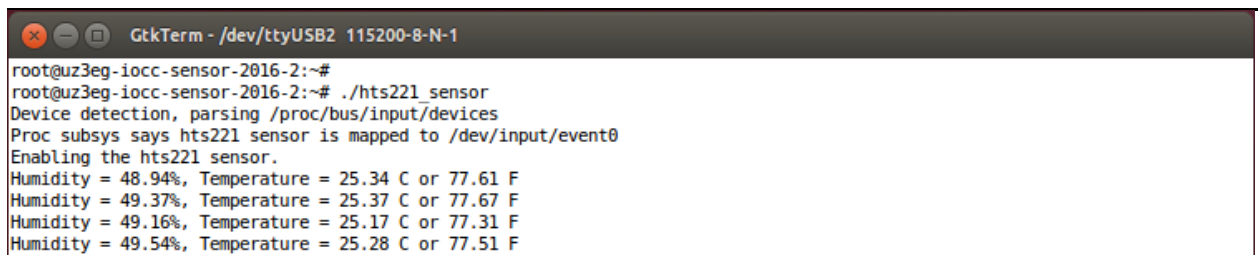
## ST Microelectronics HTS221 Thermometer and Hygrometer

The hts221\_sensor software application reads humidity and temperature data from the ST Microelectronics HTS221 sensor. There are two sensors in the HTS221 device, one provides humidity and the other provides ambient temperature.

The sample application reads data from both sensors and displays the result on the PC host serial terminal. On the UltraZed-EG Starter Kit, the I<sup>2</sup>C interface is accessed from the application code using the added hts221 I<sup>2</sup>C software driver in the PetaLinux kernel.

1. Run the application and view the temperature and humidity data.

```
./hts221_sensor
```

A screenshot of a terminal window titled 'GtkTerm - /dev/ttyUSB2 115200-8-N-1'. The terminal shows the execution of the './hts221\_sensor' command. The output includes device detection messages and four lines of sensor data, each showing humidity and temperature in both Celsius and Fahrenheit.

```
root@uz3eg-iocc-sensor-2016-2:~#
root@uz3eg-iocc-sensor-2016-2:~# ./hts221_sensor
Device detection, parsing /proc/bus/input/devices
Proc subsys says hts221 sensor is mapped to /dev/input/event0
Enabling the hts221 sensor.
Humidity = 48.94%, Temperature = 25.34 C or 77.61 F
Humidity = 49.37%, Temperature = 25.37 C or 77.67 F
Humidity = 49.16%, Temperature = 25.17 C or 77.31 F
Humidity = 49.54%, Temperature = 25.28 C or 77.51 F
```

2. Press <CTRL-C> to stop the application.

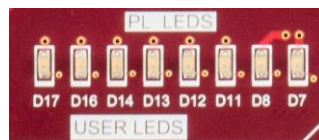
## Maxim MAX44000 Proximity Sensor

The max44000\_sensor software application reads proximity data from the Maxim Integrated MAX44000 sensor. The sample application reads data from the sensor and uses that data as input to a PWM to view the brightness of LEDs on the I/O Carrier Card. The closer an object is to the proximity sensor, the brighter the LEDs will become.

1. Run the application and wave your hand over the sensor to watch the LEDs change brightness.

```
./max44000_sensor
```

2. Notice that as the light reading from the MAX44000 sensor changes, the LEDs (D7-D14) get brighter or dimmer.



3. Press <CTRL-C> to stop the application.

## IBM Bluemix Quickstart Demo

The IIoT\_Quickstart\_Demo software application reads humidity and temperature data from the ST Microelectronics HTS221 sensor, and thermocouple temperature data from the Maxim Integrated MAX31855 sensor. The data is displayed on the PC host serial console and published to the internet via IBM Watson IoT Platform.

The application can be run in four different modes to publish data to the IB Watson cloud. The data can then be viewed at the web links shown below.

| Application Mode | Sensor Data                       |
|------------------|-----------------------------------|
| 0 (default)      | HTS221 ambient temperature        |
| 1                | HTS221 humidity                   |
| 2                | MAX31855 ambient temperature      |
| 3                | MAX31855 thermocouple temperature |

The command format is:

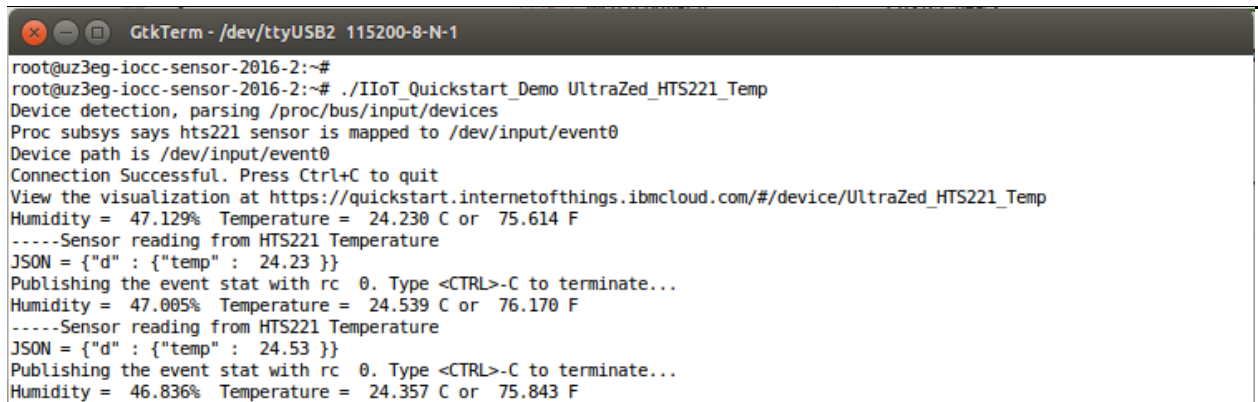
**IIoT\_Quickstart\_Demo** <unique\_identifier> <mode>

The web link format is:

**https://quickstart.internetofthings.ibmcloud.com/#/device/<unique\_identifier>**

1. Run the application and view the HTS221 ambient temperature data in the terminal window.

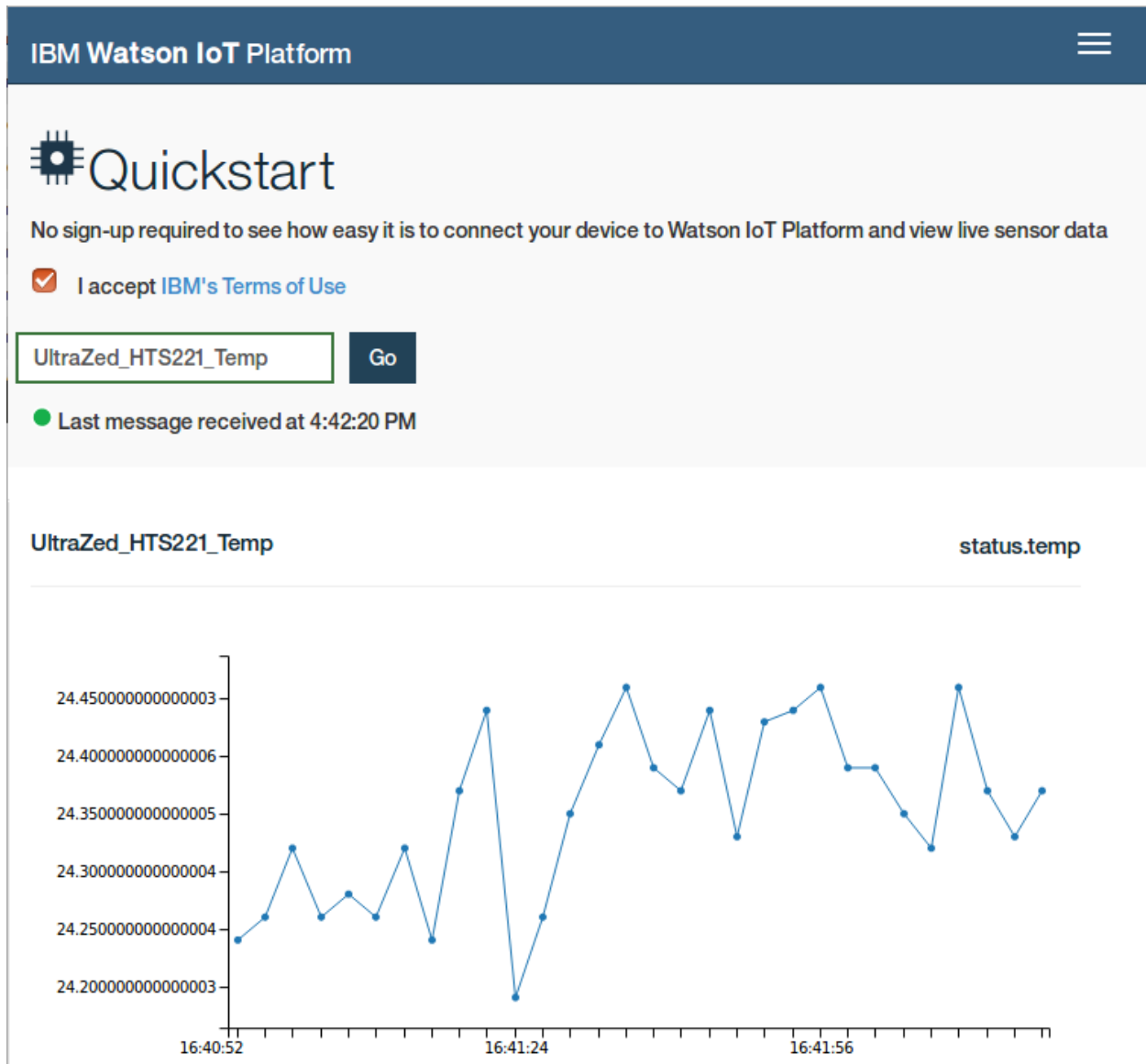
**# ./IIoT\_Quickstart\_Demo UltraZed\_HTS221\_Temp**



```
GtkTerm - /dev/ttyUSB2 115200-8-N-1
root@uz3eg-iocc-sensor-2016-2:~#
root@uz3eg-iocc-sensor-2016-2:~# ./IIoT_Quickstart_Demo UltraZed_HTS221_Temp
Device detection, parsing /proc/bus/input/devices
Proc subsys says hts221 sensor is mapped to /dev/input/event0
Device path is /dev/input/event0
Connection Successful. Press Ctrl+C to quit
View the visualization at https://quickstart.internetofthings.ibmcloud.com/#/device/UltraZed_HTS221_Temp
Humidity = 47.129% Temperature = 24.230 C or 75.614 F
-----Sensor reading from HTS221 Temperature
JSON = {"d" : {"temp" : 24.23 }}
Publishing the event stat with rc 0. Type <CTRL>-C to terminate...
Humidity = 47.005% Temperature = 24.539 C or 76.170 F
-----Sensor reading from HTS221 Temperature
JSON = {"d" : {"temp" : 24.53 }}
Publishing the event stat with rc 0. Type <CTRL>-C to terminate...
Humidity = 46.836% Temperature = 24.357 C or 75.843 F
```

2. Open the web link to view a plot of the temperature data.

[https://quickstart.internetofthings.ibmcloud.com/#/device/UltraZed\\_HTS221\\_Temp](https://quickstart.internetofthings.ibmcloud.com/#/device/UltraZed_HTS221_Temp)



3. Return to the terminal window and press <CTRL-C> to stop the application.

4. Experiment with running the application in the remaining modes. Press <CTRL-C> to stop the application when finished.
  - a. HTS221 humidity

```
./IIoT_Quickstart_Demo UltraZed_HTS221_Hum 1
```

[https://quickstart.internetofthings.ibmcloud.com/#/device/UltraZed\\_HTS221\\_Hum](https://quickstart.internetofthings.ibmcloud.com/#/device/UltraZed_HTS221_Hum)
  - b. MAX31855 ambient temperature

```
./IIoT_Quickstart_Demo UltraZed_MAX31855_Temp 2
```

[https://quickstart.internetofthings.ibmcloud.com/#/device/UltraZed\\_MAX31855\\_Temp](https://quickstart.internetofthings.ibmcloud.com/#/device/UltraZed_MAX31855_Temp)
  - c. MAX31855 thermocouple temperature. Hold the end of the thermocouple wire in your fingertips or place it near something that is hotter or colder than the ambient temperature. Notice the temperature reading change in the terminal window and the web page.

```
./IIoT_Quickstart_Demo UltraZed_MAX31855_Therm 3
```

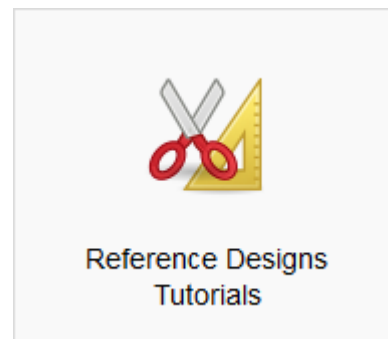
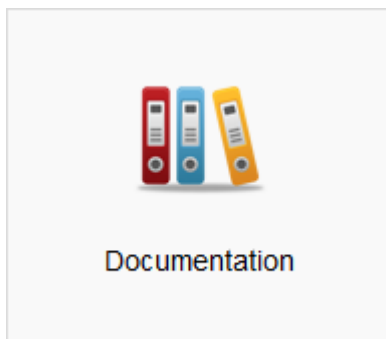
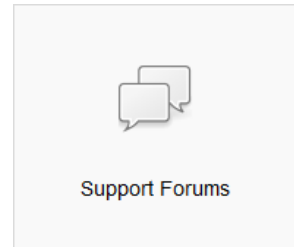
[https://quickstart.internetofthings.ibmcloud.com/#/device/UltraZed\\_MAX31855\\_Therm](https://quickstart.internetofthings.ibmcloud.com/#/device/UltraZed_MAX31855_Therm)
5. Feel free to examine the Vivado hardware platform or PetaLinux software applications for how you might adapt them to your own designs. What improvements or changes would you make?

This concludes this design tutorial.

## Appendix I: Getting Support

### Avnet Support

- Technical support is offered online through the [ultrazed.org](http://ultrazed.org) website support forums. UltraZed-EG Starter Kit users are encouraged to participate in the forums and offer help to others when possible.  
<http://ultrazed.org/forums/zed-english-forum>  
<http://ultrazed.org/forums/software-application-development>
- For questions regarding the UltraZed-EG community website, please direct questions to the ultrazed.org Web Master ([webmaster@ultrazed.org](mailto:webmaster@ultrazed.org)).
- To access the most current collateral for the UltraZed-EG Starter Kit, visit the community support page ([www.ultrazed.org/content/support](http://www.ultrazed.org/content/support)) and click one of the icons shown below:



- UltraZed-EG Starter Kit Documentation  
<http://ultrazed.org/support/documentation/17596>
- UltraZed-EG Starter Kit Reference Designs  
<http://ultrazed.org/support/design/17596/131>
- Instructions for how to setup the Ubuntu virtual machine if using a Windows host PC  
[http://ultrazed.org/sites/default/files/design/VirtualBox\\_Installation\\_Guide\\_2016\\_2.zip](http://ultrazed.org/sites/default/files/design/VirtualBox_Installation_Guide_2016_2.zip)

## Xilinx Support

For questions regarding products within the Product Entitlement Account, send an email message to the Customer Service Representative in your region:

- Canada, USA and South America - [isscs\\_cases@xilinx.com](mailto:isscs_cases@xilinx.com)
- Europe, Middle East, and Africa - [eucases@xilinx.com](mailto:eucases@xilinx.com)
- Asia Pacific including Japan - [apaccase@xilinx.com](mailto:apaccase@xilinx.com)

For technical support, including the installation and use of the product license file, contact Xilinx Online Technical Support at [www.xilinx.com/support](http://www.xilinx.com/support). The following assistance resources are also available on the website:

- Software, IP and documentation updates
- Access to technical support Web tools
- Searchable answer database with over 4,000 solutions
- User forums

## Revision History

| Date        | Version | Revision        |
|-------------|---------|-----------------|
| 25 May 2017 | 1.0     | Initial Release |
|             |         |                 |
|             |         |                 |