PSG COLLEGE OF TECHNOLOGY

Department of Electronics and Communication Engineering

Smart Home Automation using Zybo Z7-10

Main tutorial source: https://www.youtube.com/watch?v=z613J8RVpog&t=13s (materials in the description)
I completed upto 3rd step. The final step is simple but I dint have the time to do it

Contact : samuelpriyadarshan@gmail.com

Software Environment:

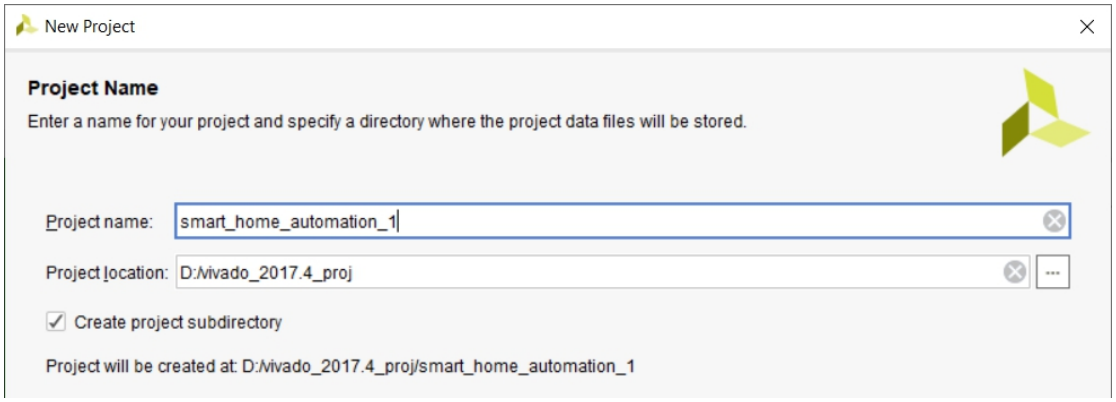
- Xilinx Vivado 2017.4
- Xilinx SDK 2017.4
- Xilinx PetaLinux 2017.4
- Ubuntu 16.04 (64-bit)

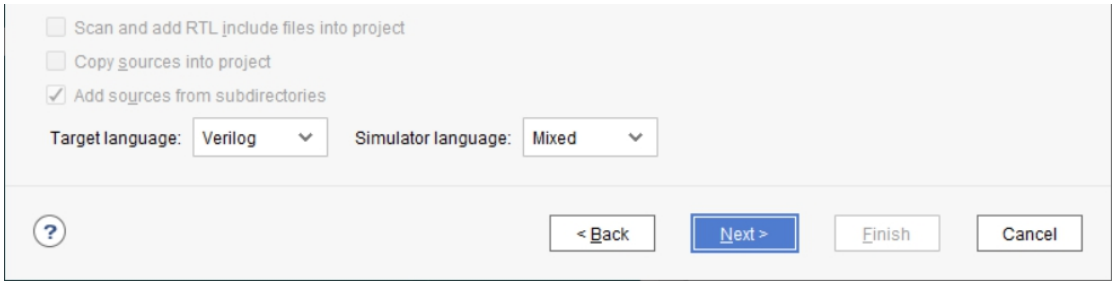
*I tried a lot of versions but the environment worked perfectly only with these exact same versions

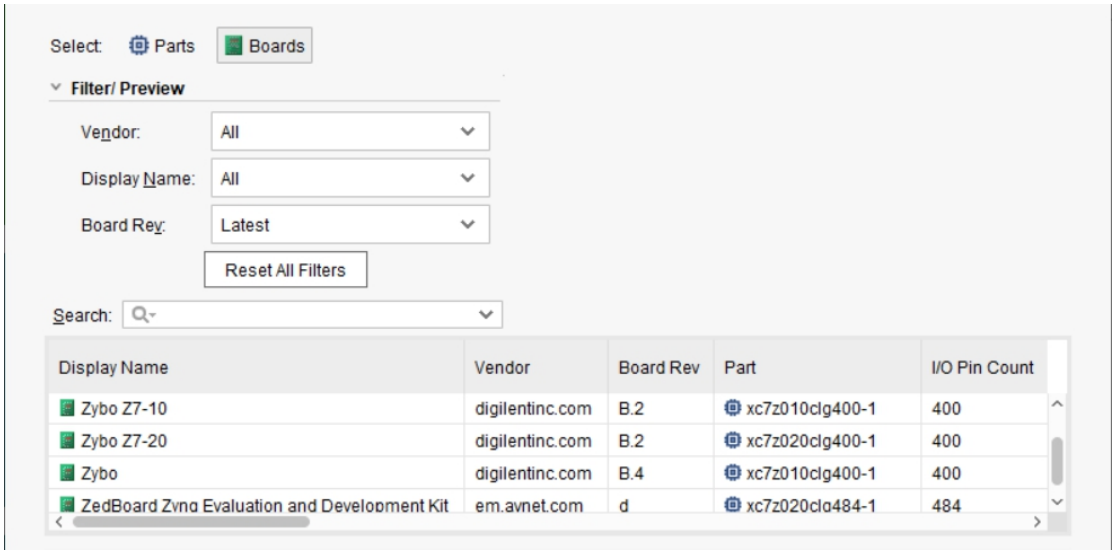
Step 1: Creating project in Xilinx Vivado 2017.4

Open Xilinx Vivado 2017.4 > Create Project > Enter project name > RTL project > Target Language : Verilog, Simulator Language : Mixed > Under default parts choose boards tab and select Zybo (board must be installed manually)* > Finish

*https://reference.digilentinc.com/reference/software/vivado/board-files

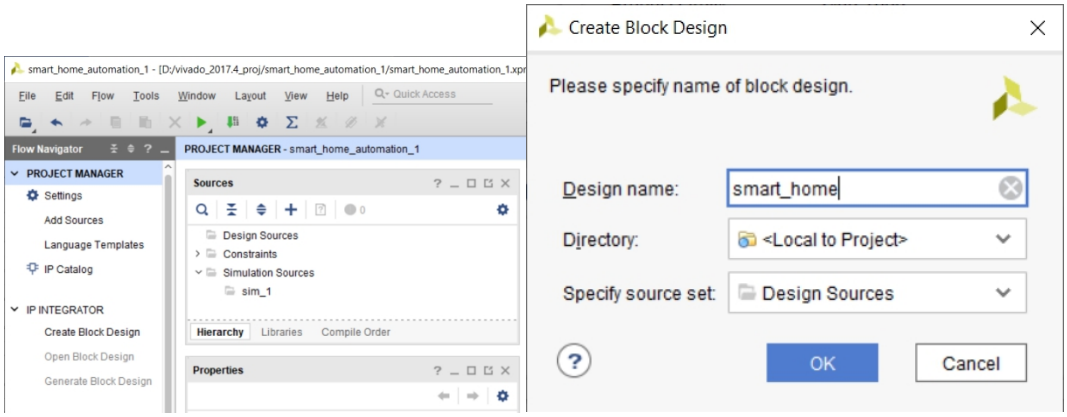


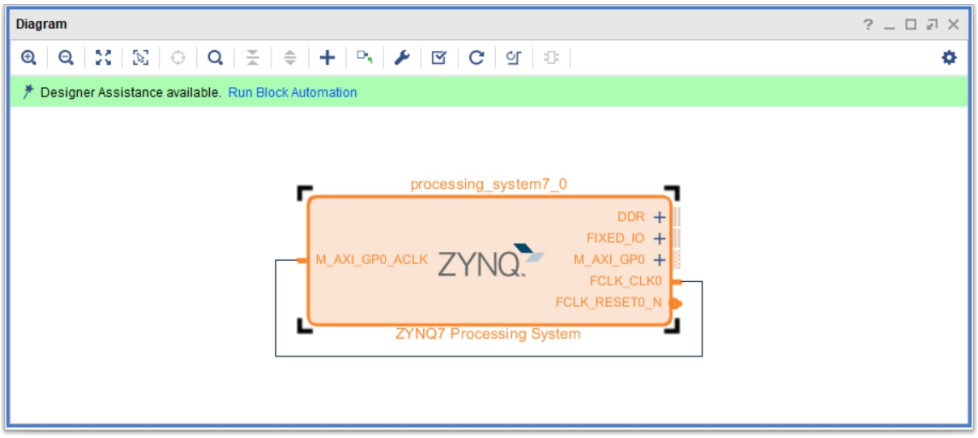


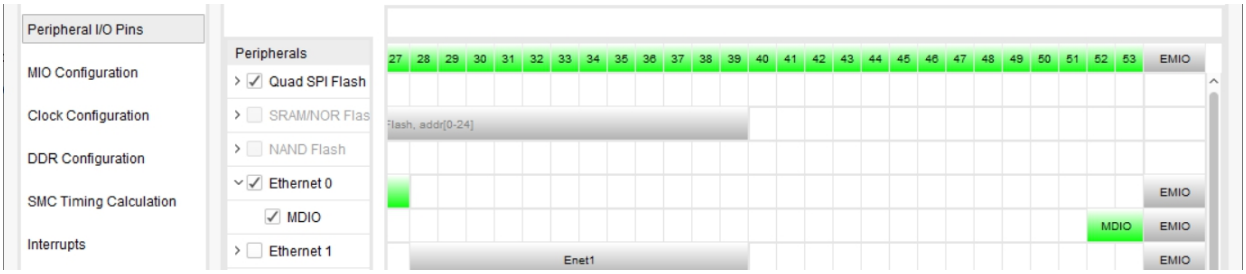


Step 2: Block design

Under IP INTEGRATOR click Create Block Design
Enter design name
Click on the + symbol or (Ctrl + I)
ZYNQ7 Processing System (Double Click)
Make a connection between FCLK_CLK0 and M_AXI_GP0_ACLK
Click Run Block Automation
(Double Click the IP)
Select Peripheral I/O Pins > Ethernet0 (Drop down arrow) > Enable MDIO
Ensure all the options in processor_1,2,3.jpg are selected (attached with the project documentation)
*However only Ethernet, SD and GPIO only will be necessary, but if you have time try only with these

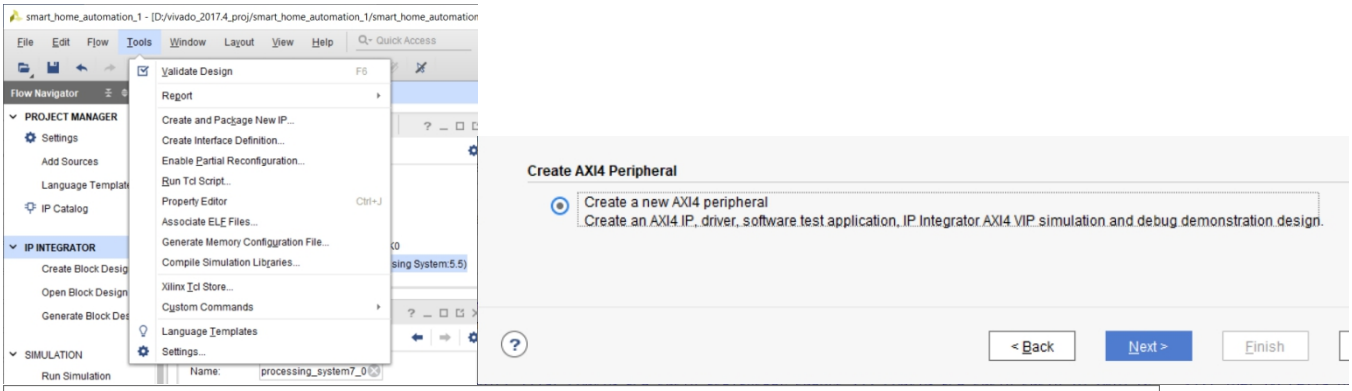


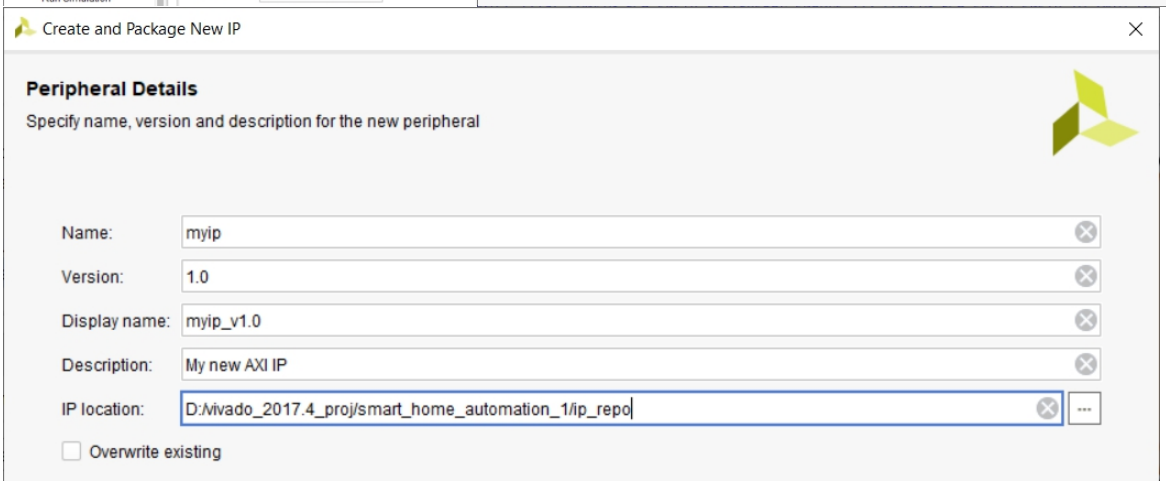


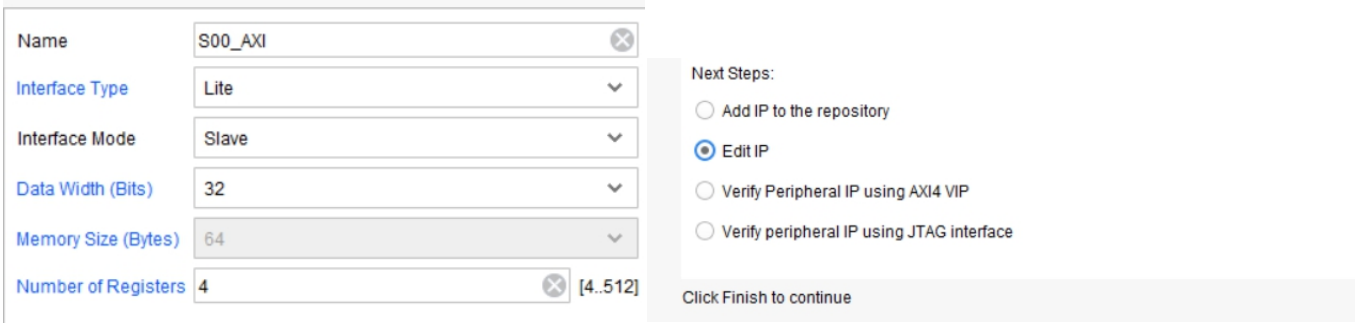


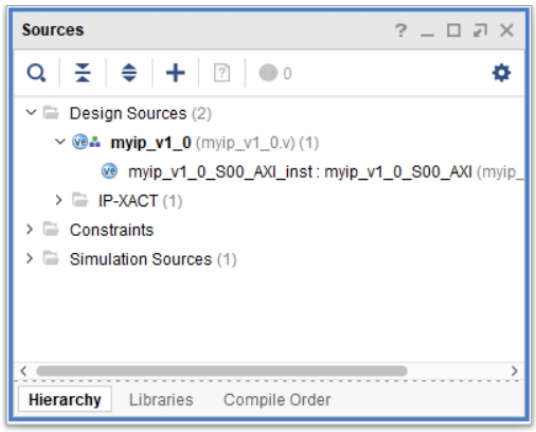
Step 3: Custom IP

Tools > Create and Package new IP > Select “Create a new AXI4 peripheral” > Enter name* > No.of Registers:4 > Select “Edit IP”.









In sources window edit the myip_v1_0.v & myip_v1_0_S00_AXI.v

myip_v1_0.v file:

Line 18: (below "// Users to add ports here")

```
output wire [3:0] LEDs,
input wire [3:0] JB_Port,
input wire [3:0] Switch,
```

After pasting

Line 53: (below "myip_v1_0_S00_AXI_inst (")

```
.LEDs(LEDs),
.JB_Port(JB_Port),
.Switch(Switch),
```

myip_v1_0_S00_AXI.v file:

Line 18: (below "// Users to add ports here")

```
output wire [3:0] LEDs,
input wire [3:0] JB_Port,
input wire [3:0] Switch,
```

After pasting

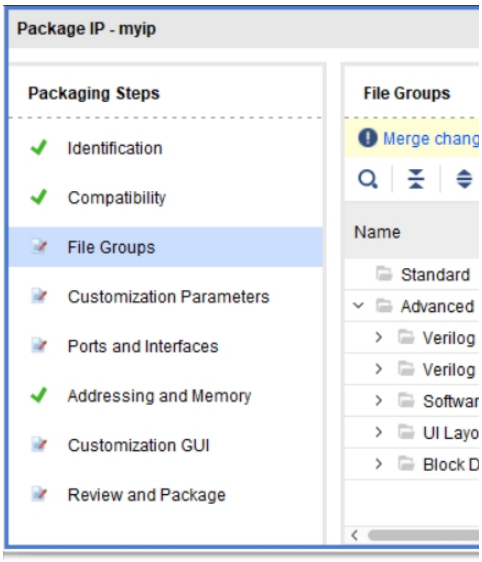
Line 403:

```
always @( posedge S_AXI_ACLK )
begin
  if ( S_AXI_ARESETN == 1'b0 )
    begin
      slv_reg1 <= 0;
      slv_reg2 <= 0;
    end
  else
    begin
```

```
            slv_reg1[3:0] <= JB_Port;
            slv_reg2[3:0] <= Switch;
          end
      end

      assign LEDs = slv_reg0[3:0];
```

After Pasting

DISABLE(type // at the starting)
Lines 226, 227, 246, 253, 264, 265.

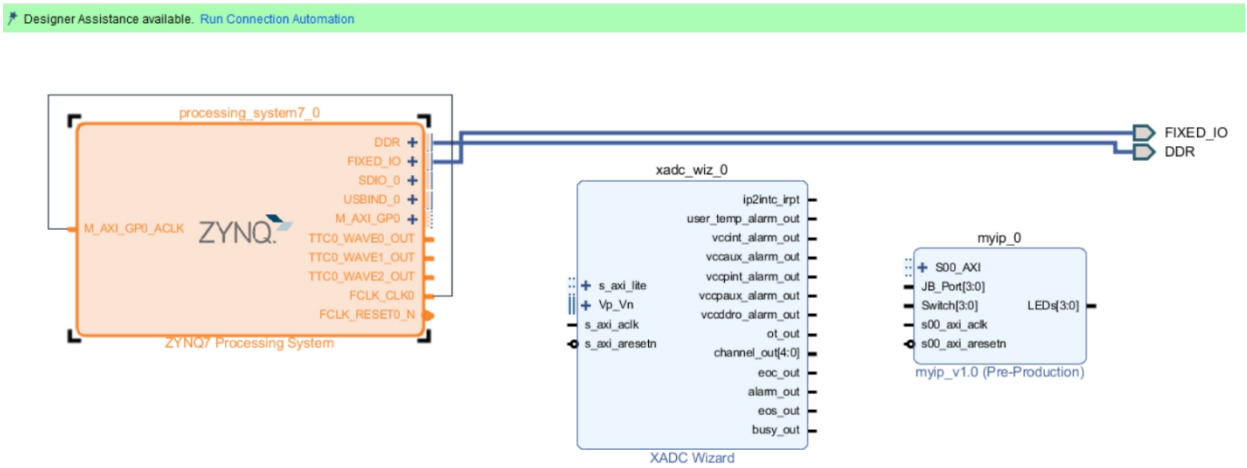


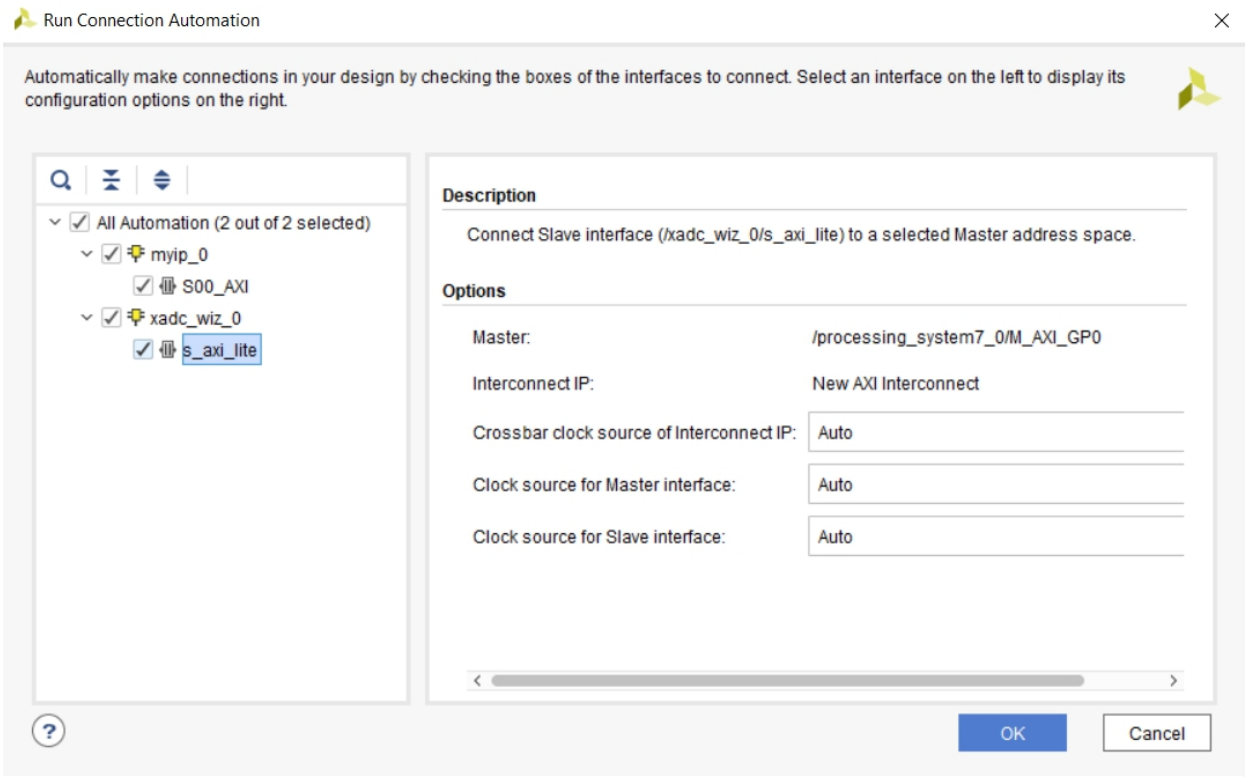
Come back to Package IP > click on the non ticked packaging steps and then click on “merge changes…” at the top

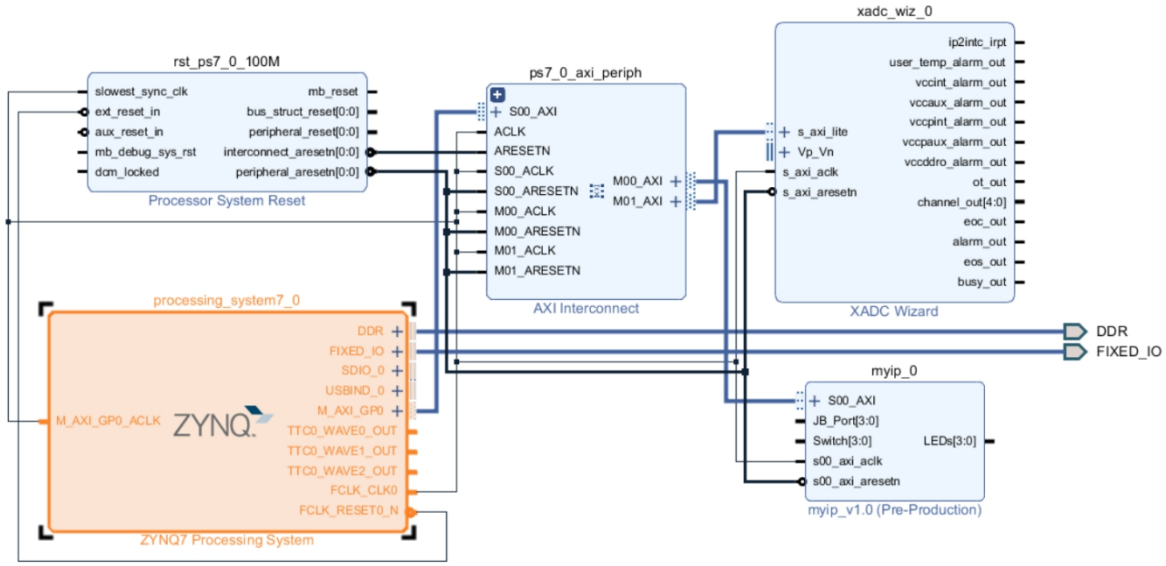
Under Review and Package > Click “IP has been modified” > Click Re-Package IP > Click yes

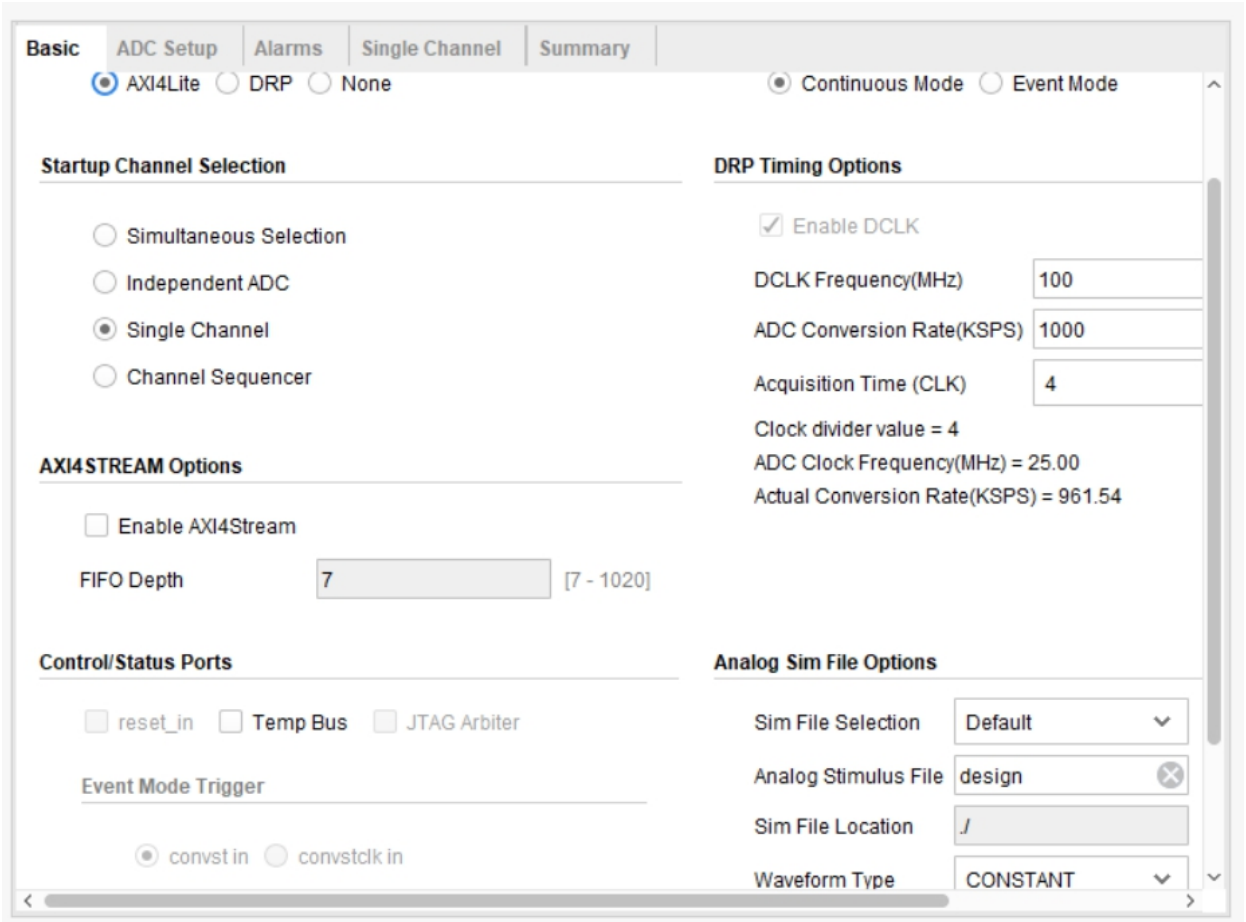
Step 4: Block Design (Contd.)

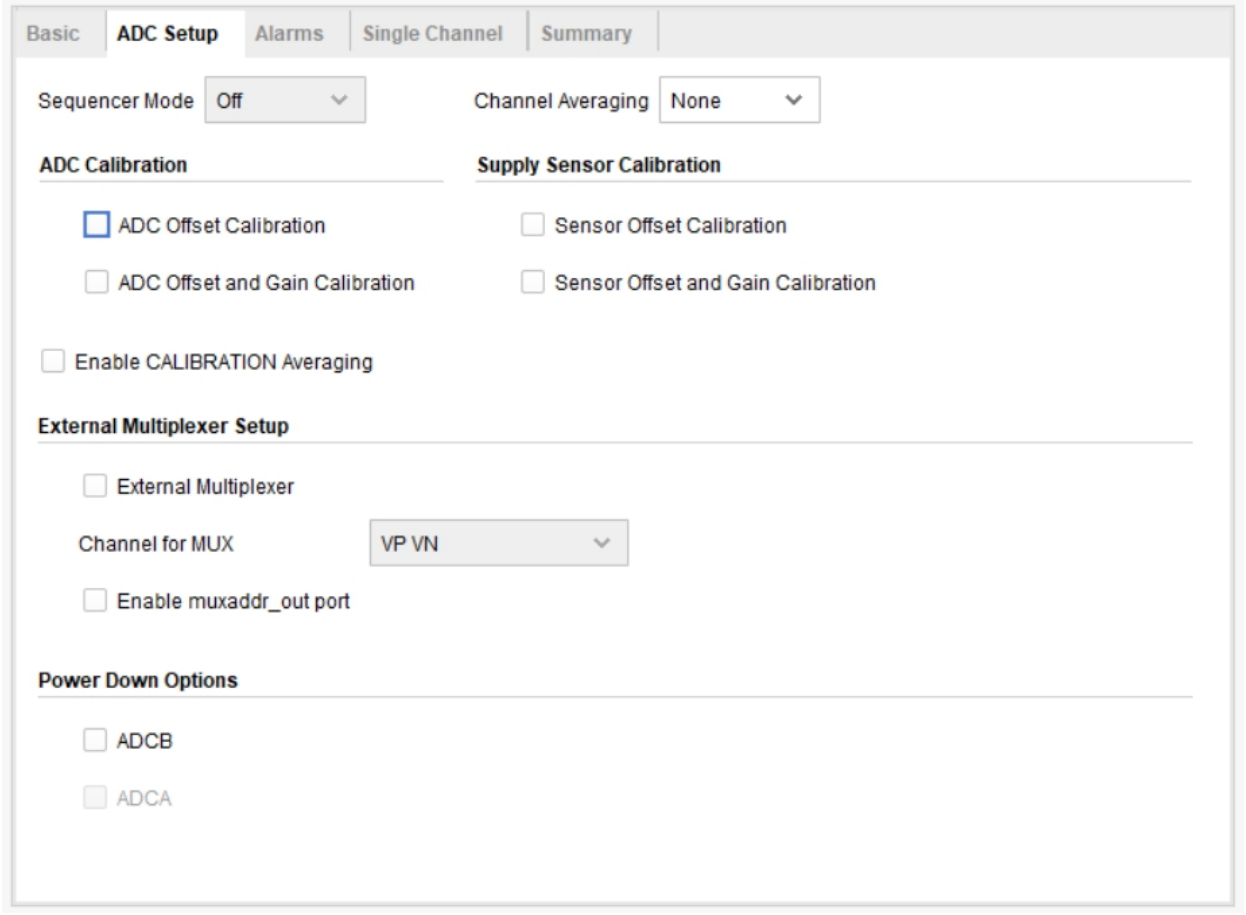
Add myip_0 and XADC Wizard (Ctrl + I) > Click “Run Connection Automation” > Enable S00_AXI & s_axi_lite > Ensure everything is set to Auto > Make External to the 3 ports in myip_0 (Select the ports and press Ctrl + T) > Double Click xadc_wiz_0 > Check images for configuration* > Click Ok > Bring the cursor near Vaux7 > Once it changes to down arrow, click it > Make External Connection for Vauxp7 and Vauxn7 > Save the design (Ctrl + S)




Run Connection Automation
Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.
All Automation (2 out of 2 selected)
myip_0
S00_AXI
xadc_wiz_0
s_axi_lite
Description
Connect Slave interface (/xadc_wiz_0/s_axi_lite) to a selected Master address space.
Options
Master:
/processing_system7_0/M_AXI_GP0
Interconnect IP:
New AXI Interconnect
Crossbar clock source of Interconnect IP:
Auto
Clock source for Master interface:
Auto
Clock source for Slave interface:
Auto
OK
Cancel


rst_ps7_0_100M
slowest_sync_clk
ext_reset_in
aux_reset_in
mb_debug_sys_rst
dcm_locked
mb_reset
bus_struct_reset[0:0]
peripheral_reset[0:0]
interconnect_aresetn[0:0]
peripheral_aresetn[0:0]
Processor System Reset
ps7_0_axi_periph
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M01_ACLK
M01_ARESETN
M00_AXI
M01_AXI
AXI Interconnect
xadc_wiz_0
s_axi_lite
Vp_Vn
s_axi_aclk
s_axi_aresetn
ip2intc_irpt
user_temp_alarm_out
vccint_alarm_out
vccaux_alarm_out
vccpint_alarm_out
vccpaux_alarm_out
vccddro_alarm_out
ot_out
channel_out[4:0]
eoc_out
alarm_out
eos_out
busy_out
XADC Wizard
processing_system7_0
DDR
FIXED_IO
SDIO_0
USBIND_0
M_AXI_GP0
M_AXI_GP0_ACLK
TTC0_WAVE0_OUT
TTC0_WAVE1_OUT
TTC0_WAVE2_OUT
FCLK_CLK0
FCLK_RESET0_N
ZYNQ7 Processing System
DDR
FIXED_IO
myip_0
S00_AXI
JB_Port[3:0]
Switch[3:0]
s00_axi_aclk
s00_axi_aresetn
LEDs[3:0]
myip_v1.0 (Pre-Production)


Basic
ADC Setup
Alarms
Single Channel
Summary
AXI4Lite
DRP
None
Continuous Mode
Event Mode
Startup Channel Selection
Simultaneous Selection
Independent ADC
Single Channel
Channel Sequencer
DRP Timing Options
Enable DCLK
DCLK Frequency(MHz)
100
ADC Conversion Rate(KSPS)
1000
Acquisition Time (CLK)
4
Clock divider value = 4
ADC Clock Frequency(MHz) = 25.00
Actual Conversion Rate(KSPS) = 961.54
AXI4 STREAM Options
Enable AXI4Stream
FIFO Depth
7
[7 - 1020]
Control/Status Ports
reset_in
Temp Bus
JTAG Arbiter
Event Mode Trigger
convst in
convstclk in
Analog Sim File Options
Sim File Selection
Default
Analog Stimulus File
design
Sim File Location
./
Waveform Type
CONSTANT


Basic
ADC Setup
Alarms
Single Channel
Summary
Sequencer Mode
Off
Channel Averaging
None
ADC Calibration
ADC Offset Calibration
ADC Offset and Gain Calibration
Supply Sensor Calibration
Sensor Offset Calibration
Sensor Offset and Gain Calibration
Enable CALIBRATION Averaging
External Multiplexer Setup
External Multiplexer
Channel for MUX
VP VN
Enable muxaddr_out port
Power Down Options
ADCB
ADCA

Basic | ADC Setup | **Alarms** | Single Channel | Summary

Over Temperature Alarm (°C)

Trigger 125.0 [-40.0 - 125.0]

Reset 70.0 [-40.0 - 125.0]

User Temperature Alarm (°C)

Trigger 85.0 [-40.0 - 125.0]

Reset 60.0 [-40.0 - 125.0]

VCCINT Alarm (Volts)

Lower 0.97 [0.0 - 1.05]

Upper 1.03 [0.0 - 1.05]

VCCAUX Alarm (Volts)

Lower 1.75 [0.0 - 1.89]

Upper 1.89 [0.0 - 1.89]

VCCBRAM Alarm (Volts)

Lower 0.95 [0.0 - 1.05]

Upper 1.05 [0.0 - 1.05]

VCCPint Alarm (Volts)

Lower 0.95 [0.0 - 1.05]

Upper 1.00 [0.0 - 1.05]

VCCPaux Alarm (Volts)

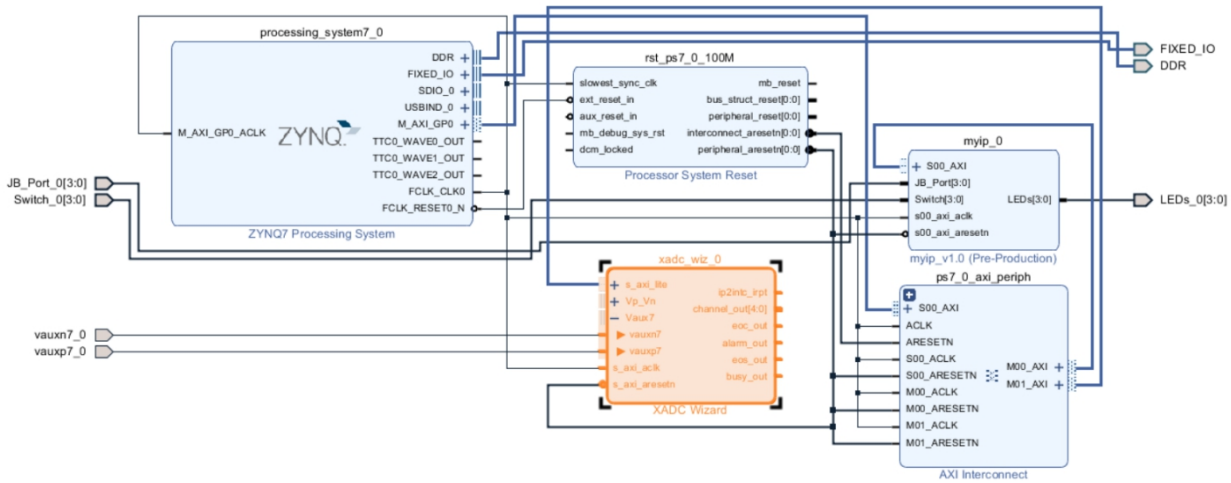
Lower 1.71 [0.0 - 1.89]

VCCDdro Alarm(Volts)

VCCDDRO Voltage

Basic | ADC Setup | Alarms | **Single Channel** | Summary

Select Channel	Channel Enable	Average Enable	Bipolar	Acquisition Time
VAUXP7 VAUXN7	☑		☐	☐



Step 5: Generate Bitstream

Under Source tab >Click the + Symbol > Select Add or Create Constraints > Click Create File > Copy the Constrint file from folder and save > Under Design Sources > Right Click smart_home.bd > Click Create HDL Wrapper > Ignore the message > Under flow navigator - PROGRAM AND DEBUG - Click Generate Bitstream

Once bitstream is successfully generated, Click on File > Export > Export hardware > Check Include bitstream > Click OK
The hardware file is saved in ".hdf" format in smart_home_automation_1.sdk folder (subject to change if u go for a different project name) under the project folder. Copy this file and paste it in the shared folder of Ubuntu.

Step 6: Generating Linux Kernels

OS Platform: Ubuntu 16.04

Open Terminal and type the following commands

cd ~/opt/pkg/petalinux/source settings.sh
mkdir project
cd project
mkdir smart_home_automation_1

(you can make the file directory according to your convenience. When you are working on multiple designs, its better to name them accordingly)

cd smart_home_automation_1
petalinux-create --type project --template zynq --name PjIot

(the --name can be changed according to your convenience. I named it as PjIot)

Copy the .hdf file from the shared folder and paste it into project/smart_home_automation_1/PjIot
Type the following commands in terminal

petalinux-config --get-hw-description

(Shows a new menu, but you need not change and cofiguration for this project)

In files go to project/smart_home_automation_1/PjIot/project-spec/meta-user/recipes-bsp/device-tree/file
In this location you'll find a system-user.dtsi file. Either replace this file with the files files provided or type it out manually*. The file sets the custom IP as a UIO device and sets the Vaux7 pin.

*changing indentation can give you an error.

petalinux-config -c rootfs (Can take a long time to run)

(opens a new window)
Filesystem Packages → devel → python → python (Select all the files in it using 'Y')
Filesystem Packages → misc → gcc-runtime-> libstdc ++

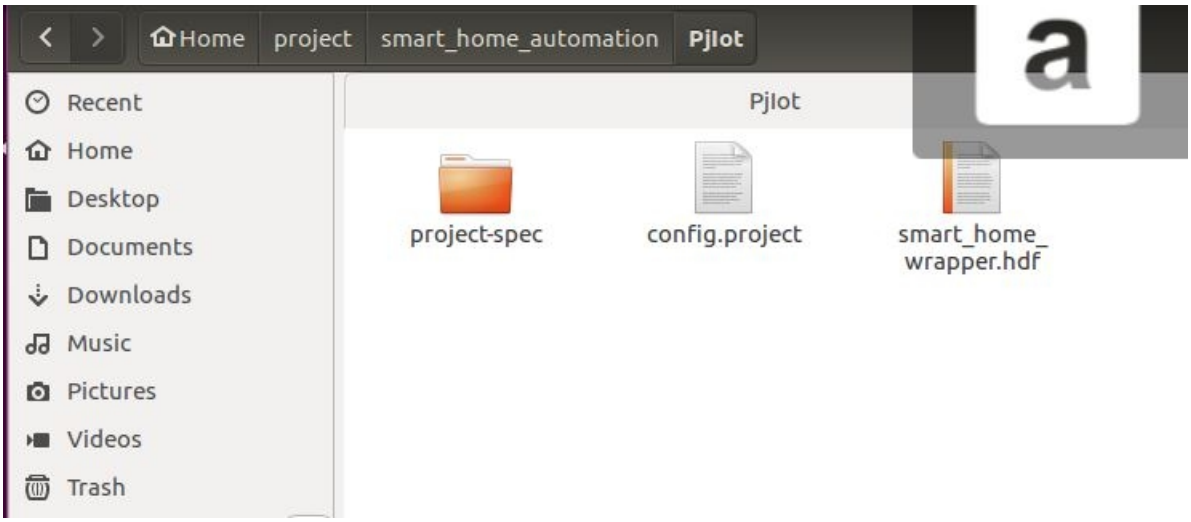
(save on exit)

petalinux-build (Can take a long time to run)
petalinux-package --boot --force --fsbl images/linux/zynq_fsbl.elf --fpga images/linux/smart_home_wrapper.bit --u-boot

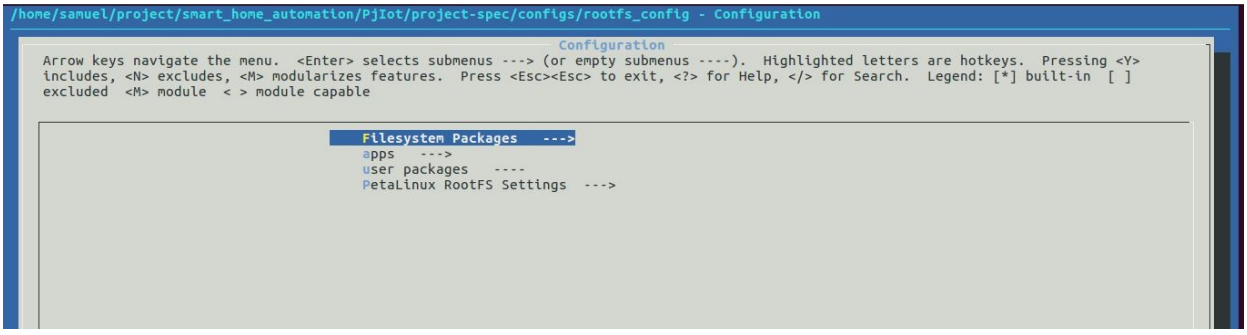
Go to the project/smart_home_automation_1/PjIot/images/linux

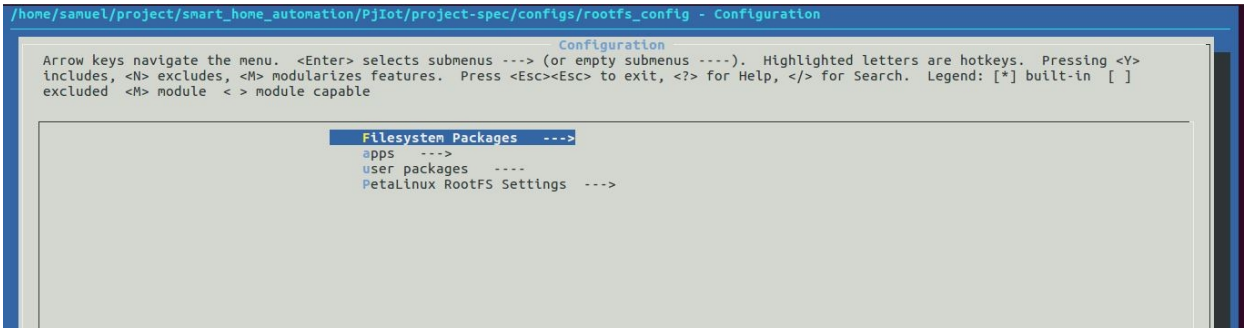
Copy and paste BOOT.BIN and image.ub file into the shared folder

```
samuel@samuel-VirtualBox:~$ cd ~/opt/pkg/petalinux
samuel@samuel-VirtualBox:~/opt/pkg/petalinux$ source settings.sh
PetaLinux environment set to '/home/samuel/opt/pkg/petalinux'
WARNING: /bin/sh is not bash!
bash is PetaLinux recommended shell. Please set your default shell to bash.
INFO: Checking free disk space
INFO: Checking installed tools
INFO: Checking installed development libraries
INFO: Checking network and other services
WARNING: No tftp server found - please refer to "PetaLinux SDK Installation Guide" for its impact and solution
samuel@samuel-VirtualBox:~/project/smart_home_automation$ petalinux-create --type project --name PjIot --template zynq
INFO: Create project: PjIot
INFO: New project successfully created in /home/samuel/project/smart_home_automation/PjIot
```



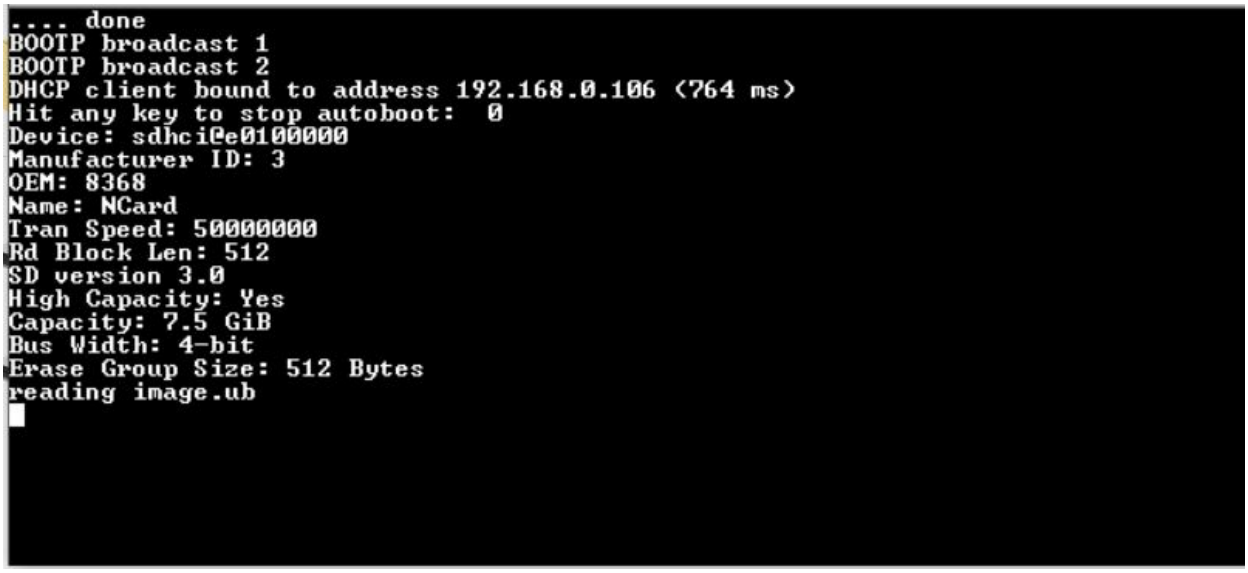

```
samuel@samuel-VirtualBox:~/project/smart_home_automation$ cd PjIot
samuel@samuel-VirtualBox:~/project/smart_home_automation/PjIot$ petalinux-config --get-hw-description
INFO: Getting hardware description...
cp: omitting directory '/home/samuel/project/smart_home_automation/PjIot/build'
cp: omitting directory '/home/samuel/project/smart_home_automation/PjIot/components'
cp: omitting directory '/home/samuel/project/smart_home_automation/PjIot/project-spec'
INFO: Rename smart_home_wrapper.hdf to system.hdf
[INFO] generating Kconfig for project

[INFO] menuconfig project
/home/samuel/project/smart_home_automation/PjIot/build/misc/config/Kconfig.syshw:30:warning: defaults for choice values not supported
/home/samuel/project/smart_home_automation/PjIot/build/misc/config/Kconfig:568:warning: config symbol defined without type
configuration written to /home/samuel/project/smart_home_automation/PjIot/project-spec/configs/config

*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.

[INFO] sourcing bitbake
[INFO] generating plnxtool conf
[INFO] generating meta-plnx-generated layer
~/project/smart_home_automation/PjIot/build/misc/plnx-generated ~/project/smart_home_automation/PjIot
~/project/smart_home_automation/PjIot
[INFO] generating machine configuration
[INFO] generating bbappends for project . This may take time !
~/project/smart_home_automation/PjIot/build/misc/plnx-generated ~/project/smart_home_automation/PjIot
~/project/smart_home_automation/PjIot
[INFO] generating u-boot configuration files

[INFO] generating kernel configuration files
[INFO] generating kconfig for Rootfs
Generate rootfs kconfig
[INFO] oldconfig rootfs
[INFO] generating petalinux-user-image.bb
```

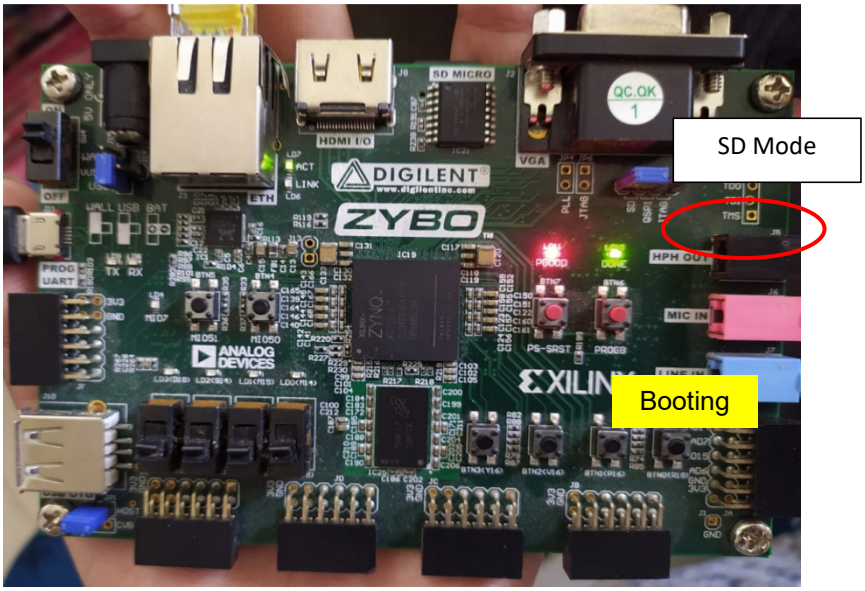




STEP 7: SD Card and booting using Serial Port monitor'

Copy the BOOT.BIN and image.ub file into a formatted SD Card. Note that the SD Card is FAT32, which is mostly the format of any SD Card from 4GB memory or above. Once it is copied, connect the ethernet cable into the Ethernet port of Zybo Board and turn the board ON (Do not insert SD Card now). Make sure that the Modem/Router is DHCP enabled. Also, ensure that the jumper near VGA port is set to SD mode . Open Serial Port Monitor in PC (I used tera term). You can use any kind of serial port monitor like tera term, PuTty etc. On opening Tera Term, it should show a window with TCF Agent and Port option. If your zybo board is connected, it will show the respective COM port and make sure to enable the Serial Option. You must be able to find the COM port of Zybo in Device manager of windows. Next, click on Setup -> Serial Port from the task bar. A new window appears. Ensure that the following parameters are set accordingly. Once it is done, turn OFF the board and then insert SD Card and now turn the board ON again. You should see the Board booting successfully.





Step 8: Xilinx SDK

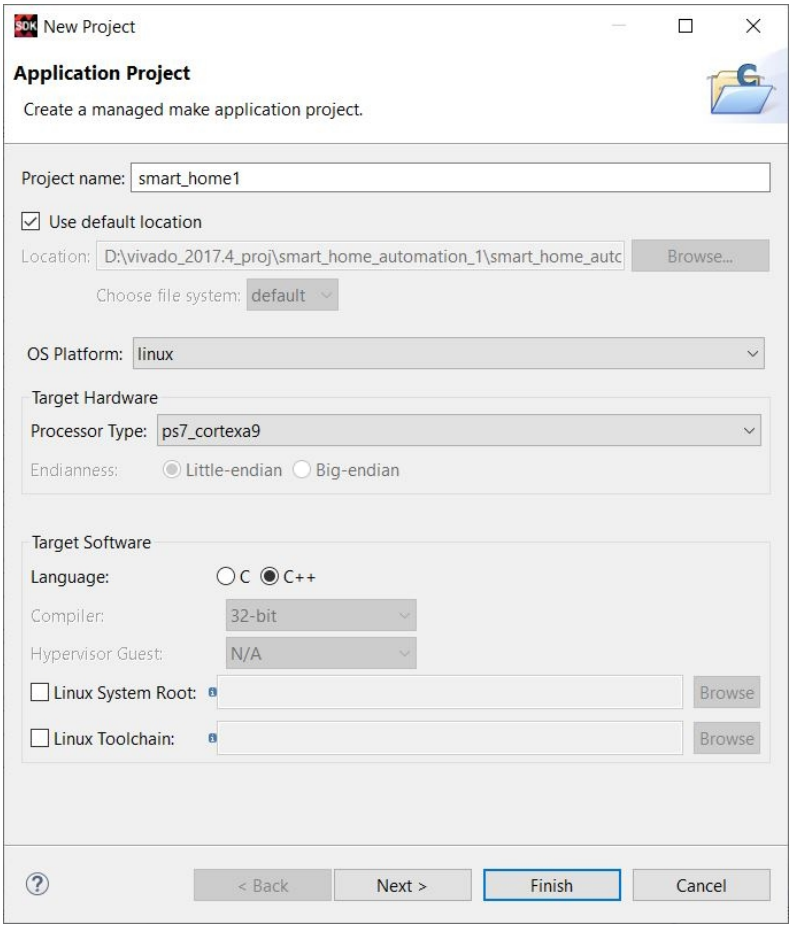
Once the SD card has successfully booted, enter login and password
In this case both of them are root. Once you have come inside the root folder, create two fifo files with 666 mode as it helps in communicating between python and CPP.
For this, type the following commands in the serial monitor

```
root@PjIot:~#
mkfifo pipe_c2p
root@PjIot:~#
root@PjIot:~# mkfifo pipe_p2c
root@PjIot:~#
root@PjIot:~# chmod 666 pipe_p2c
root@PjIot:~#
root@PjIot:~# chmod 666 pipe_c2p
root@PjIot:~#
root@PjIot:~#
```

Once these commands are typed they pave the way for making a connection between Python and CPP. Also type “ifconfig” to get the IP address of the zybo board as it will be necessary in later stages.

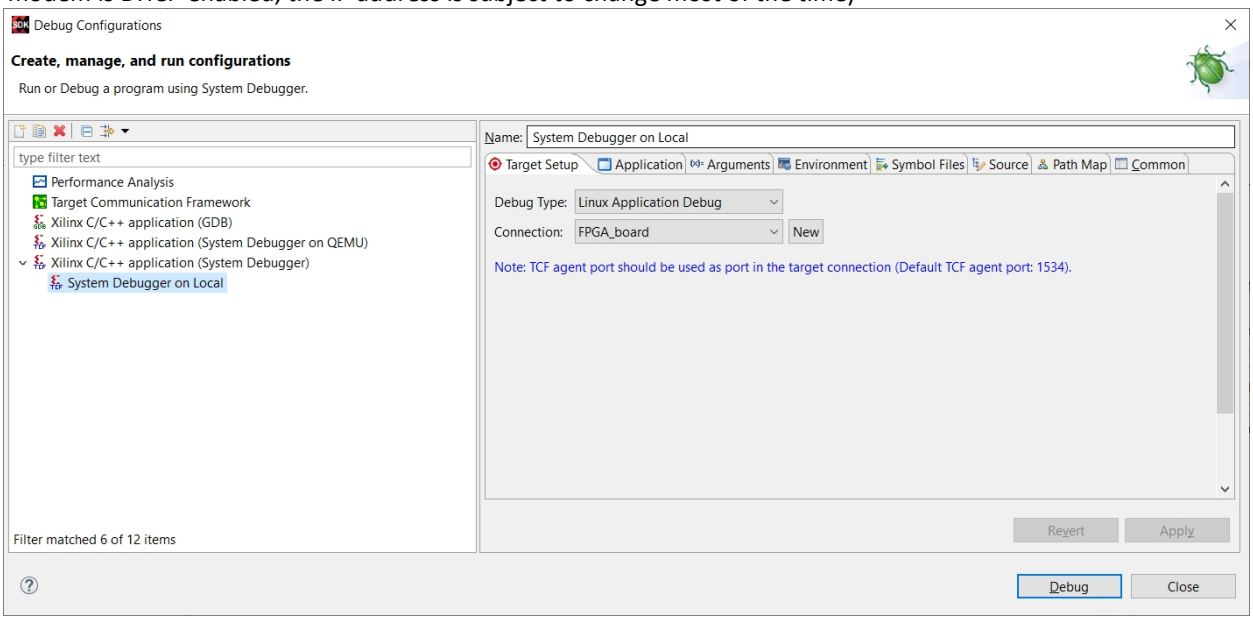
Next Open the project in xilinx > click on File > Launch SDK. This procedure will direct you to the workspace in Xilinx SDK. If you wish to do this directly by opening Xilinx SDK, mention the workspace and the hdf file and you’ll be good to go.

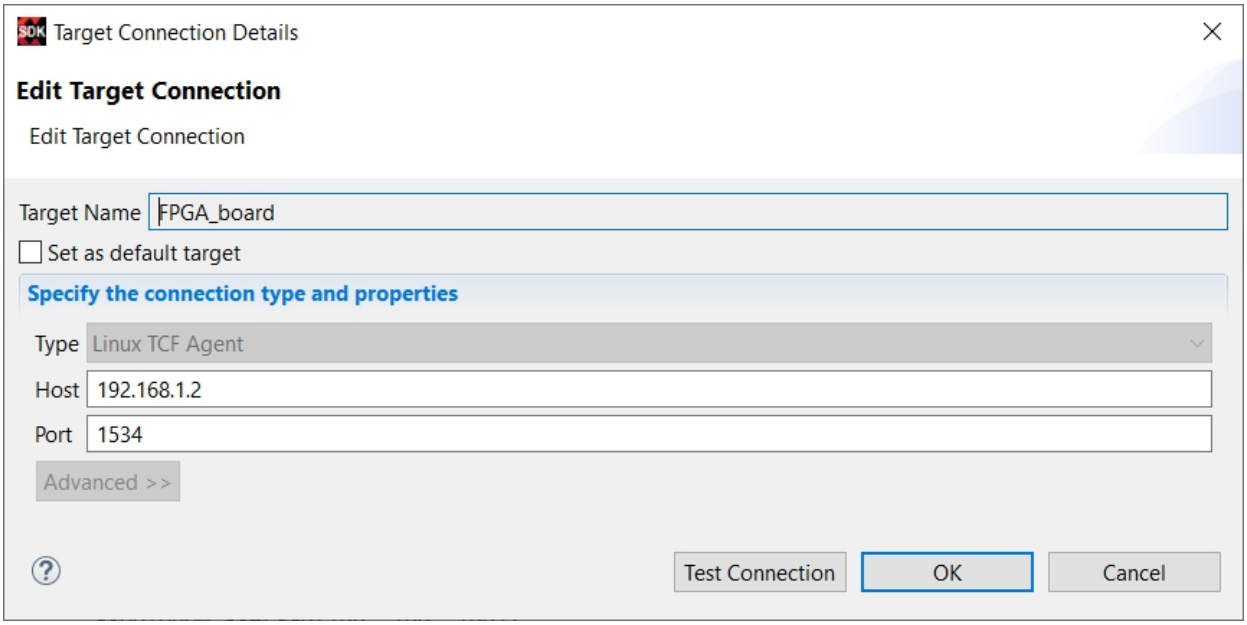
Once you’re inside the work space, click on file > new > Application project (Alt+Shift+N)
Give a project name of your choice (for this project ‘smart_home’)> OS platform select linux > Language : C++ > Click Next > Choose Empty Application and click on Finish



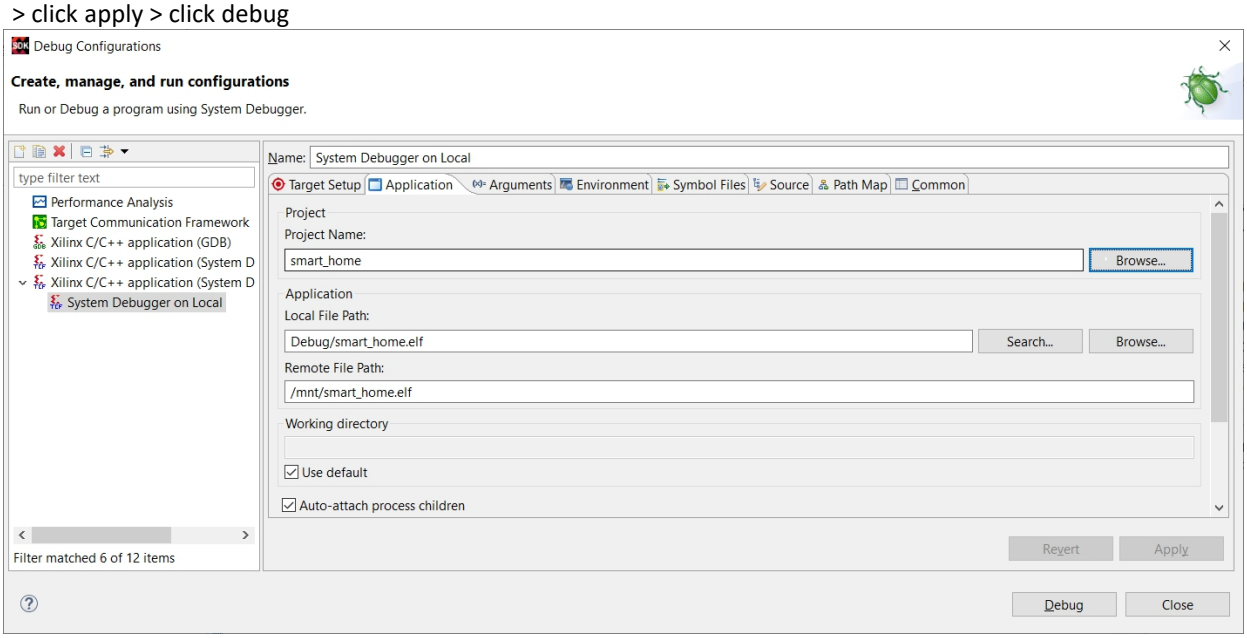
Click the dropdown next to ‘smart_home’ > src > replace this folder by src folder attached with this documentation
Click on Project in the toolbar > Clean > Clean all projects > Click OK (This builds the project. Most probably it wouldnt give any error)

Click on Run in the Toolbar > debug configuration > Xilinx C/C++ application (System Debugger) > Under target setup tab > Debug Type: Linux Application Debug > Click on New next to connection (may differ from screenshot) > Enter a Target name > Host: The IP address of your board > Click test connection > It should give a success message (Make sure your laptop or PC is connected to the same network as that of your Zybo board. I used an old wireless router so that it can be accessed wirelessly. As the modem is DHCP enabled, the IP address is subject to change most of the time)





Next, under Application tab next to target setup > Click on browse under project name > Select smart_home(your project name) > click apply > click debug

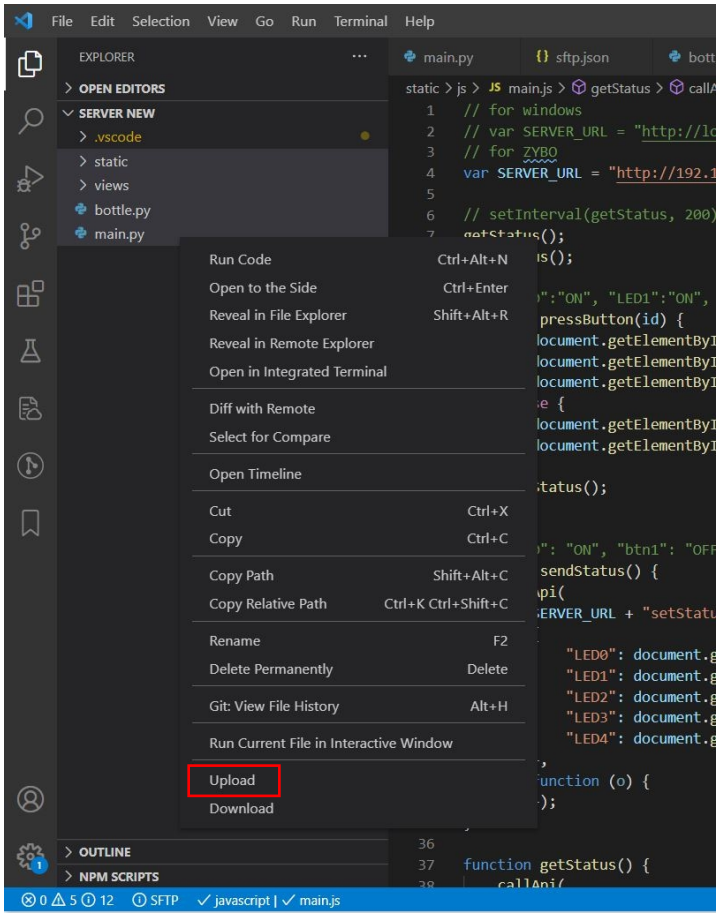


This will take you to the debug window. The debugger will start automatically. If it doesnt start, right click on the system debugger and select relaunch.

Step 9 : Python application development using Visual Studio

Once the debug starts running, open VS Code > File > Open folder > Smart home server (attached with the project documantation)

Open command pallete (Ctrl + Shift + P) > type SFTP: Config (enable SFTP in Extension menu) > This will open a sftp.json file in the project folder > Copy the contents from .vscode/sftp.json file or just replace it entirely (make sure you enter your IP address in python, sftp.json and views/main.js files)> Select all the static, views, main.py and bottle.py file > right-click and select upload
Check if the files are uploaded to your board by typing ‘ls’ in the serial monitor



Now in the serial port monitor > type python main.py (this will start the server)
Open browser > type 192.168.1.2:8080 (Enter your IP address and add 8080) > the web page should work