



**Driver development for Zedboard peripherals and their
integration into the SOC LINUX operating system
and
Connecting ZedBoard with Wifi through ESP8266**

201614032 Aleyna KARAPINAR

201614042 Fatmanur ÖZEL

Supervisor: Assoc. Prof. Orhan GAZİ

Report Date: 02.06.2021

Abstract

The final achievement of this project is to develop and implement a custom and Embedded Linux Operating System (OS) integrated with a specific PL peripheral. This OS will be developed on ZedBoard (Zynq Evaluation & Development Board) development kit, Xilinx's Zynq-7000 All Programmable System on Chip which contains a dual core ARM Cortex-A9 and a 7 Series FPGA Artix-7. [1] After performing these operations in the first term, we continued our second term project by providing communication between Zedboard and ESP8266 Wifi module.

Özet

Bu projenin amacı, belirli bir PL çevre birimi ile entegreli, Gömülü Linux İşletim Sistemi (OS) geliştirmek, uygulamaktır ve UART üzerinden data transferi yapmaktır. Bu işletim sistemi, ZedBoard (Zynq Değerlendirme ve Geliştirme Kurulu) geliştirme kiti, Xilinx'in çift çekirdekli ARM Cortex-A9 ve 7 Serisi FPGA Artix-7 içeren Zynq-7000 üzerinde geliştirilecektir. [1] İlk aşamada bu işlemleri yaptıktan sonra projemize Zedboard ile ESP8266 Wifi modülü arasındaki iletişimi sağlayarak devam ettik.

1. Introduction

Nowadays Embedded System based applications are increasing. Its very usual to find them. Inside a television, a refrigerator, a car or even inside a plane. These things are very different but not the tools to develop the Embedded System. The best characteristic of an Embedded System is its versatility because with the same computer and the same tools it is possible to develop systems completely different. The Embedded Systems have the same advantages of software tools in the electronic systems field, allowing the designer to develop specific applications independently of the hardware architecture in an easy way.

There are many options to build an Embedded System regarding to the operating system. Some of those options are Android OS, Embedded Windows or Embedded Linux. The Embedded Linux is the mostly used in the professional environment for that reason it will be the option chosen to develop the example of this document.

There are several ways to build an embedded system but all of them share some common steps. One of these steps is the Linux customization, for this purpose Linuxlink will be used.

Linuxlink is a tool created by Timesys Company to build custom configurations of Embedded Linux in an easy way. Another purpose of this document is to understand how to use this tool and build an example project with it.

Another issue to solve is the hardware platform where the Embedded Linux is going to be used. There are many different platforms and not all are supported by Linuxlink. The development platform chosen is the Zedboard, based on Xilinx's Zynq 7000 System on Chip which contains a dual core ARM Cortex-A9 and a 7 Series FPGA Artix-7. This is not the most complete platform but allows designer to start in this field without many problems and is a good approach to professional platforms.[2]

Another topic is mentioned that UART. In UART communication, two UARTs communicate directly with each other. The transmitting UART converts parallel data from a controlling device like a CPU into serial form, transmits it in serial to the receiving UART, which then converts the serial data back into parallel data for the receiving device. Only two wires are needed to transmit data between two UARTs.

Data flows from the Tx pin of the transmitting UART to the Rx pin of the receiving UART. UARTs transmit data asynchronously, which means there is no clock signal to synchronize the output of bits from the transmitting UART to the sampling of bits by the receiving UART. Instead of a clock signal, the transmitting UART adds start and stop bits to the data packet being transferred. These bits define the beginning and end of the data packet so the receiving UART knows when to start reading the bits.

When the receiving UART detects a start bit, it starts to read the incoming bits at a specific frequency known as the baud rate. Baud rate is a measure of the speed of data transfer, expressed in bits per second (bps). Both UARTs must operate at about the same baud rate. The baud rate between the transmitting and receiving UARTs can only differ by about 10% before the timing of bits gets too far off. [3]

1.1 Document Structure

The document is divided into three main sections:

The first section contains the definitions of everything involved in this project. At the same time, this section is divided into five parts:

In first place an Embedded Linux definition.

Second, Zedboard development platform definition.

The third part explains the tools involved in the building process. Every tool has different purpose and theses purposes will be described.

In fourth place, an alternative to Linuxlink will be presented.

Finally, all main files involved in the process will be described to understand the specific function of each one.

The second part is divided into four parts:

The first part describes the Linuxlink building process.

The second one describes the Vivado design cycle.

In third place the SDK process will be described.

The last part describes some other actions needed for booting the system.

The last section is divided into four main sections:

First of all, the board to be used was selected.

Then the processing system is selected and the uart connections are removed.

Third, the SDK was migrated and the necessary codes were embedded in the card.

Finally, the data received via UART was displayed using the Teraterm program.[4]

2. Requirements

Why Develop and Implement an Embedded Linux Operating System:

There are a countless number of reasons to develop and implement an embedded Linux system. The most relevant will be explained during this section and are summarize below.

- Community support and possibility of taking part into it.

- Devices coverage.

- Eases the new features testing.

- Full control.

- Low cost.

- Platform reusage.

- Quality.

- RTOS (Real Time Operating System) possibility.

After knowing the major advantages of embedded Linux systems, it will be perfectly clear the usefulness of knowing how to develop and implement a system of this kind.

Community Support and Possibility of Taking Part into It:

Linux is an open-source code; therefore, there are a great number of developers and user communities sharing their knowledge and code. This results in a high-quality support in which anyone can directly contact with many developers who are working or have been working in the same topic.

In addition, these communities are usually internet communities, allowing 24-hour availability to the user and speeding up the problems resolution.

Finally, there is also the opportunity of taking part into the different communities, for example to bug report; to add new code, versions or patches; etc.

- Devices Coverage

Due to the rest of its advantages, there are a great range of systems based on embedded Linux kernel, such as smartphones, tablets, PDAs, smart TVs, machine control, and medical instrument, among others.

- Eases the New Features Testing

As already mentioned, Linux is an open-source code. Therefore, it is really easy to get a piece of software and evaluate it. It allows studying several options while making a choice. Furthermore, new possibilities and solutions can be investigated.

As a result, it is too much easier and cheaper than purchasing or using proprietary-product trial versions.

- Full Control

The developer can have access to the source code for all components, allowing unlimited changes, modifications, and optimizations without vendor lock-in. Therefore, the developer has full control over the software.

Nevertheless, that is not the case of proprietary embedded operating systems, where the opposite is the case.

- Low Cost

Being an open-source includes being free of charge. Therefore, this free software can be duplicated on as many devices as it was necessary with no costs.

It is one of the key advantages, and it can be considered that all other benefits have been produced as a consequence of this advantage.

- Platform Re-usage

Linux already provides many components and code for standard well-know functions, such as libraries, multimedia, graphics, protocols, etc.

It allows quickly developing complex products, based on easier, already available components. Therefore, it is not necessary to re-develop the same code by different developers.

Being able to re-use the components and code is another of the key advantages of embedded Linux. It results from the rest of advantages of embedded Linux over proprietary embedded operating systems.

- Quality

The open-source components are widely used, in a great multitude of systems. Therefore, a large number of users develop different embedded Linux components and share their knowledge, allowing designing a high quality system with high quality components.

- RTOS (Real Time Operating System) Possibility

Another benefit of using an embedded Linux RTOS over a traditional proprietary RTOS is that the Linux community tends to support new IP and other protocols faster than RTOS vendors do. [5]

3. Materials and Methods

Zedboard development platform:

Zedboard was created to be a community development platform evaluation and development board based on Xilinx Zynq-7000 All Programmable System on Chip. Combining a dual Cortex- A9 Processing System with 85000 7-Series Programmable Logic cells, the board allows the designer to implement many different applications and use many different connections. Features of zedboard is shown in Figure 1.

- **Zynq Processor**
 - Xilinx XC7Z020-1CGL484CES Zynq-7000 AP SoC
 - Upto 667MHz operation
 - NEON™ Processing / FPU Engines
- **Audio**
 - 24-bit stereo audio CODEC
 - Stereo line in/out
 - Headphone
 - Microphone input
- **Connectivity**
 - 10/100/1000 Ethernet
 - USB OTG (Device/Host/OTG)
 - USB UART
- **Debug/Programming**
 - On-board USB JTAG programming port
 - ARM Debug Access Port (DAP)
- **Video/Display**
 - HDMI output (1080p60 + audio)
 - VGA connector
 - 128 x 32 OLED
 - User LEDs (9)
- **Memory**
 - 512 MB DDR3 memory (1066 Mbps)
 - 256 Mb Quad SPI Flash
 - Full size SD/MMC card cage
 - 4 GB SD Card Included
- **Dimensions**
 - Length: 16cm
 - Width: 13.5cm
- **Expansion**
 - FMC (Low Pin Count)
 - (5) Pmod™ headers (2x6)
- **Certification**
 - CE and RoHS certified
- **Power**
 - 12V DC input @ 3.0 A (Max)

Figure 1

Figure 2 shows the peripherals that are included on Zedboard and a block diagram that explains how they are inter-connected.

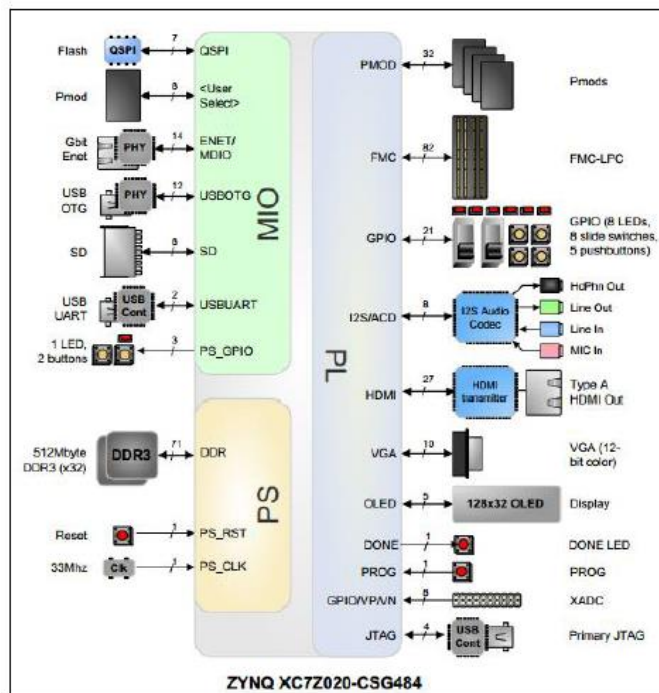


Figure 2 – Zedboard Block Diagram

Xilinx Zynq-7000 Soc

The heart of the Zedboard is a Xilinx All Programmable SoC architecture. This Soc integrates a dual-core ARM Cortex-A9 MPCore based Processing System (PS) and a Xilinx Programmable Logic (PL) in a single chip. The PS includes on-chip memory, external memory interfaces and a set of I/O peripherals.

Figure 3 illustrates the functional blocks of the Zynq-7000 AP SoC. PS and PL are on separated power domains, allowing the user to power down the PL management.

The processors in the PS usually boot first. The PL could be configured as part of the boot process or configured after. The PL can be completely reconfigured or used with partial, dynamic reconfiguration (PR). PR allows the configuration of a part of the PL.

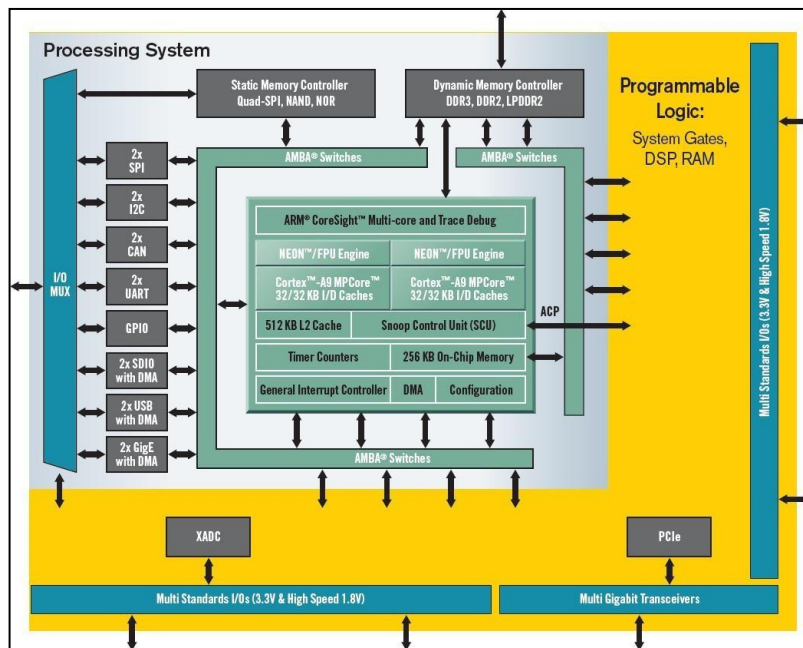


Figure 3 – Zynq 7000 functional blocks

Tools

There are three tools involved in the process:

Linuxlink:

Linuxlink is an online tool, developed by Timesys Company that allows the designer to create an Embedded Linux distribution by choosing the main features as the kernel, root file system (RFS), toolchain and packages to install into the RFS. It is not a completely online tool because it could be installed in a computer but this tool needs the internet connectivity in order to do some checks about the updates and to download the new packages if chosen.

Vivado:

Vivado is a Xilinx Company tool. Its purpose is to configure the hardware. With this tool the interconnections between the PS and the PL are configured. It is possible to add customize hardware such as specific purpose peripherals and many other things.

SDK:

SDK is a Xilinx Company tool. Its purpose is to help the designer with the software. It has the possibility to generate very important software as the first stage bootloader (FSBL), device tree blob (DTB) and BOOT.bin among others.[6]

Tera Term: It is an open-source, free, software implemented, terminal emulator (communications) program. It emulates different types of computer terminals, from DEC VT100 to DEC VT382. It supports telnet, SSH 1 & 2 and serial port connections. It also has a built-in macro scripting language (supporting Oniguruma regular expressions) and a few other useful plugins.[7]

Main files definition

First at all the definition of “main file” will be shown.

Main file: it is not a common definition, it is only used to distinguish which files are important and which of them are very important. In this case the very important files will be defined. This means, the files that are going to be inside the SD card or the files that are strictly needed to generate the previous ones.

The files are not displayed in alphabetical order instead they are displayed in order of appearance. They are:

uImage: this is the kernel image. This file is configurable. This file is the first of the four main files.

Device Tree Blob (dtb): It contains the hardware defined in Vivado as PS and PL. It could be modified by turning it into the device tree source (dts) file which is a text file. This is the second of the four main files.

BOOT.bin: This is a file generated from the output.bif file. To understand the third main file lets going to see the output.bif first:

Output.bif: this file contains the description of three files. This file contains some instructions to create the BOOT.bin. In it is specified if the files referenced are bootloader, data, etc.

The referenced files in output.bif are:

FSBL.elf:

This file implements a first boot where the minimal configurations are done such as configure the UART. Its name means First Stage Boot Loader. It is generated by SDK software.

bitstream.bit: it contains the Programmable Logic (PL) configuration, this is an optional file and it is generated by Vivado software.

u-boot.elf: this file boots the custom Embedded Linux.

uramdisk.image.gz: this is the RFS image. It contains the packages selected for the Embedded Linux distribution. This file will be decompressed when the system boots. This is the fourth of the four main files.

Custom Distribution

In this chapter an embedded system will be developed, from the beginning to the system booting. Also a test user program and a debugging will be shown.

The building process for a minimal embedded system is divided in four main steps:

The first step includes the selection of the kernel, the device tree blob, the root files system and the u-boot files.

The second step includes the creation of the bitstream and the Processing System configuration. Also the FSBL and BOOT files will be generated.

The third step describes the additional actions needed to boot the system correctly.

The last step prepares the host for booting the system and boots it.

Figure 4 describes the complete process.

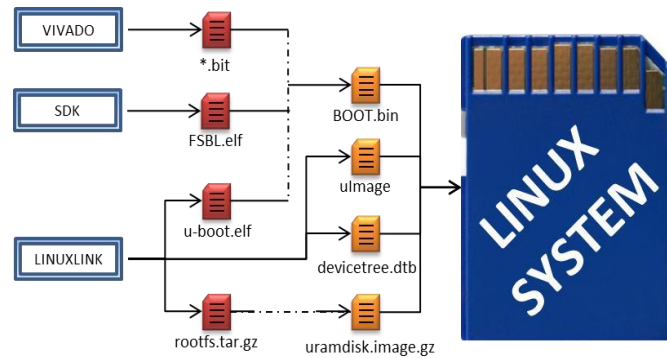


Figure 4 – Custom Distribution Process

Figure 5 shows the main steps of the Linuxlink process. The files in red are needed to create the final files. The files in orange will be copied into the SD card

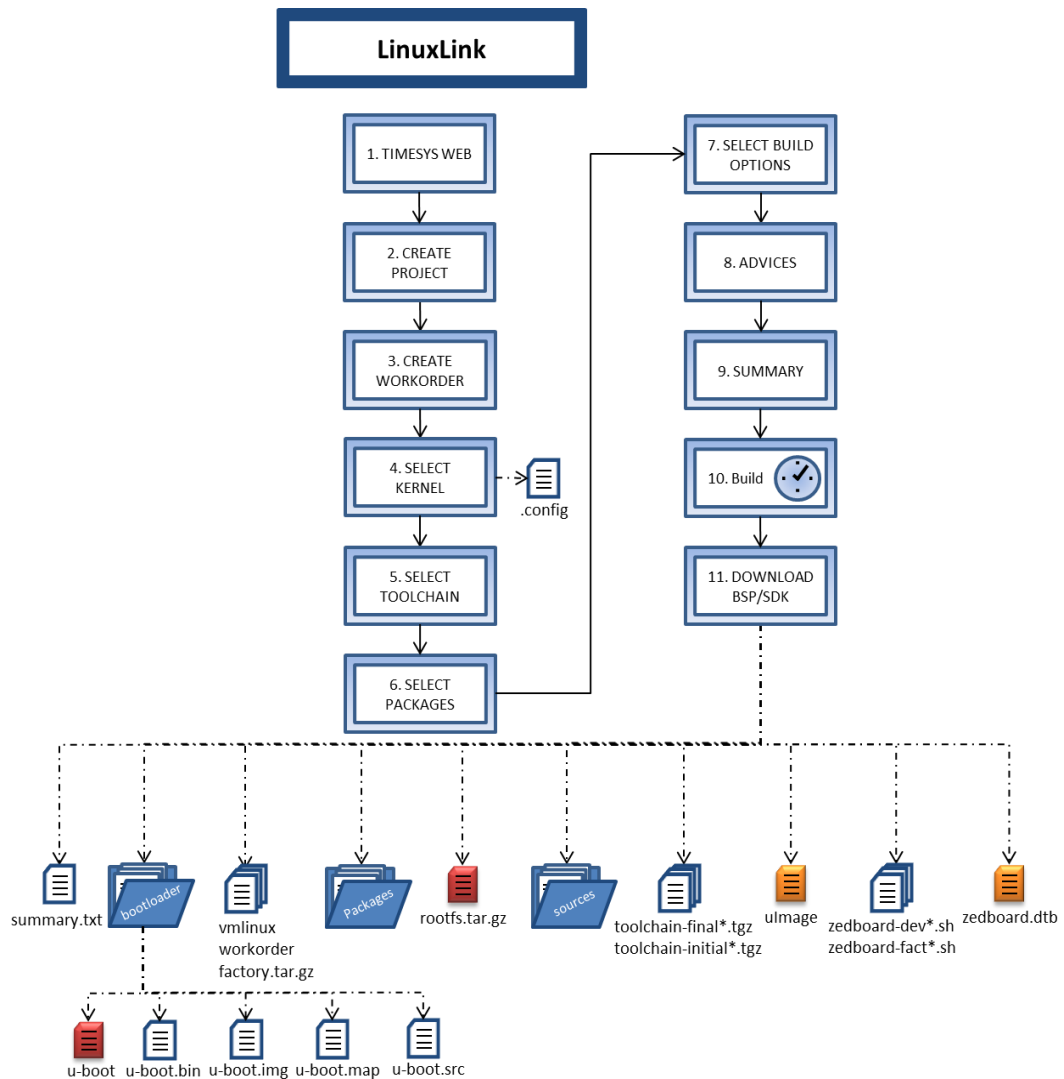


Figure 5 – Linuxlink Design Flow

Figure 6 is described the de main steps of the design flow for Vivado and SDK software. Files represented in red are going to be used to generate an orange file that is one of the main files for booting the system. Notice that u-boot.elf was generated with Linuxlink.

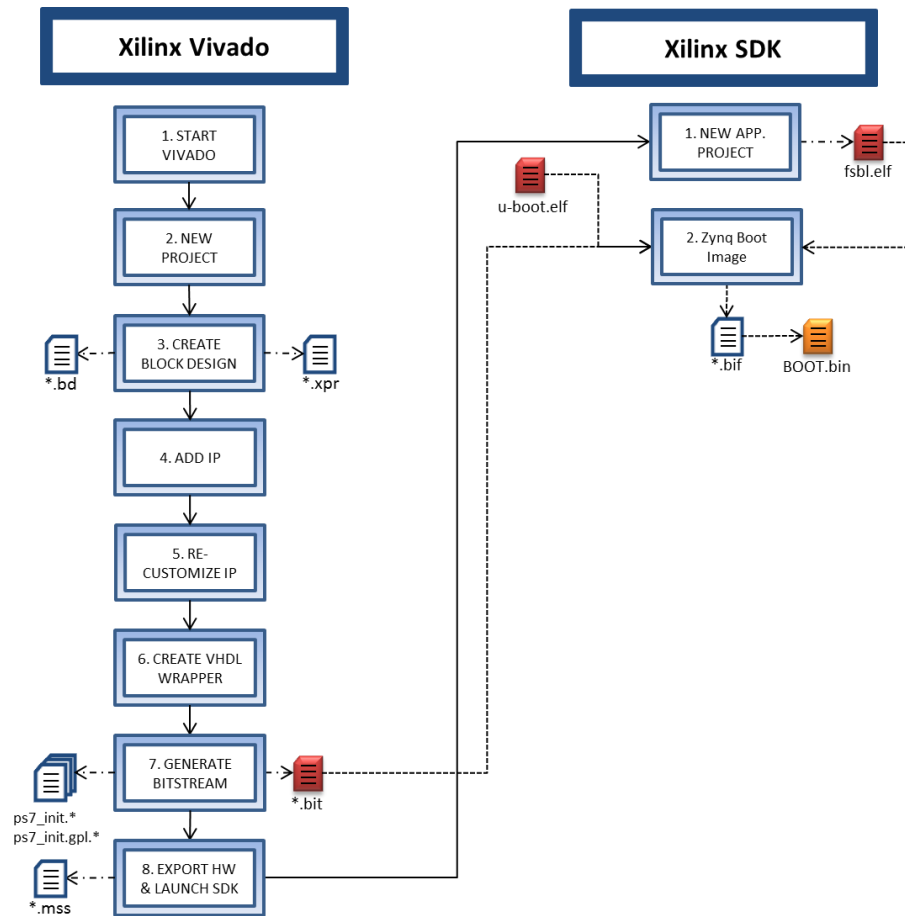


Figure 6 – Vivado & SDK Design Flow

The script execution

At this point there are already three files ready to copy into the SD card, only one remains, this is the “uramdisk.image.gz” or the RFS

The file generated by Linuxlink cannot operate as RFS so it needs to be converted. Additionally some changes in the RFS could be done. This is shown in Figure 7.

Be sure that “rootfs.tar.gz” and “rfs.sh” are in the same folder. Run the script with “./rfs.sh”.

Set the RFS size. It could not be bigger than 10MB for this example.

Decide if a “getty” is needed. Getty is an utility that prompts a message at the end of the booting process to ask the user for a login.

Set a root password.

Make every additional changes needed for the system and at the end finish the conversion. Finally, “uramdisk.image.gz” file has been created.

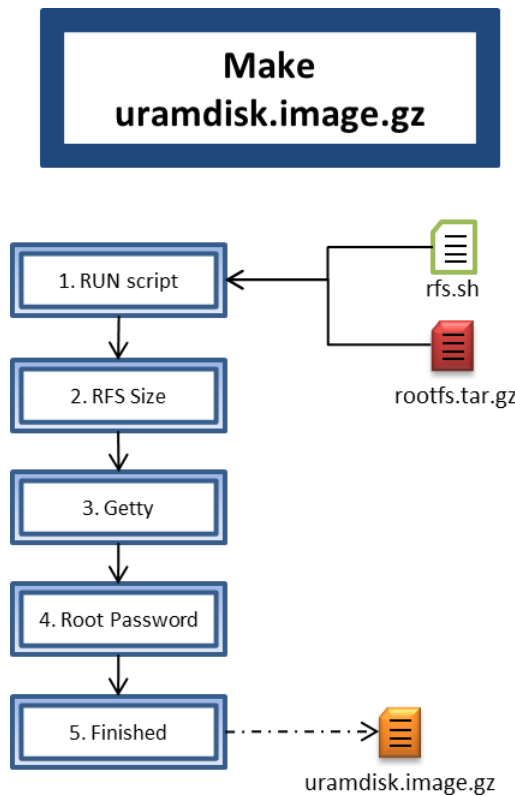


Figure 7 – Uramdisk build

Booting the system

First of all, the booting process, depicted in figure 8,

BootROM: Stage 0 Boot Loader, located in BootRom, will start to run. The boot mode pins are checked to know from where to load the FSBL.

Boot.bin is loaded by Stage 0 Boot Loader. FSBL starts to run and downloads the bitstream file to the PL system.

After that, FSBL loads the u-boot to begin loading the operating system.

U-boot calls the autoboot procedure to check the BootMode pins again for the source of the kernel image. The procedure sdboot starts.

Sdboot reads the kernel image, device tree blob and the RFS. U-boot starts to run the kernel image from where sdboot loaded it. The root file system is loaded by the kernel. Finally the login prompt appears.

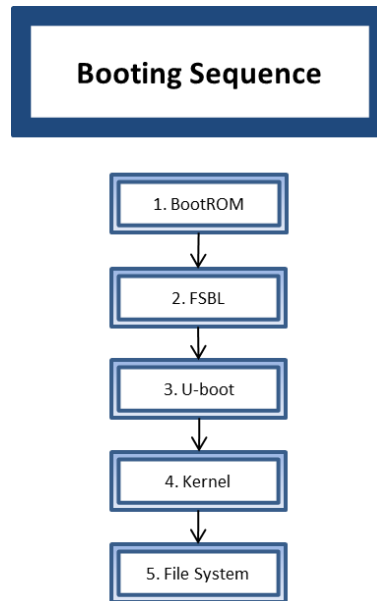


Figure 8 – Booting Sequence

4. Hardware or Software Design

The block diagram created to add an operating system to Zedboard is shown in figure 9.

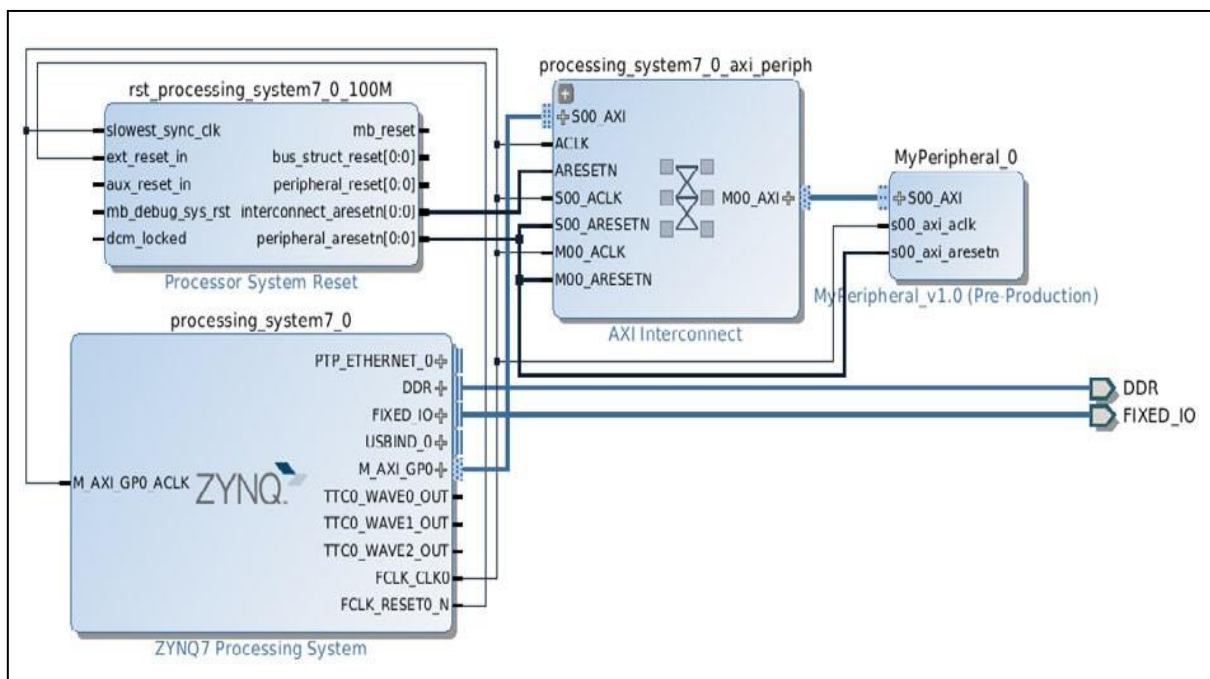


Figure 9 - Vivado Block Diagram

The connection between ESP8266 WIFI module and Zedboard is shown in figure 10. [8]

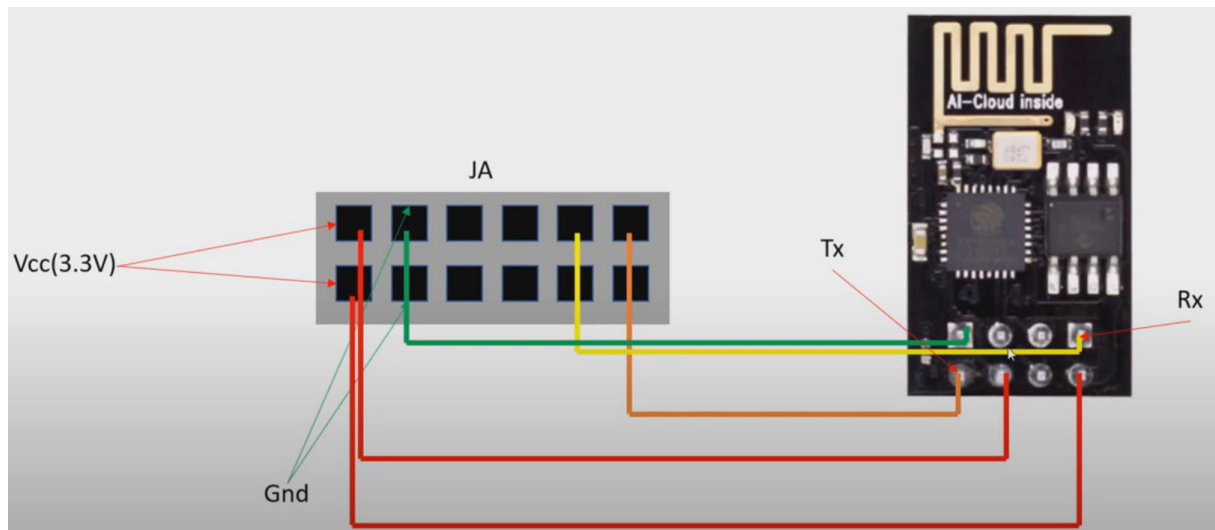


Figure 10 - UART Block Diagram

5. Test conditions

We used the oscilloscope whether the data came to the wifi module and this part is shown in figures 11 and 12.

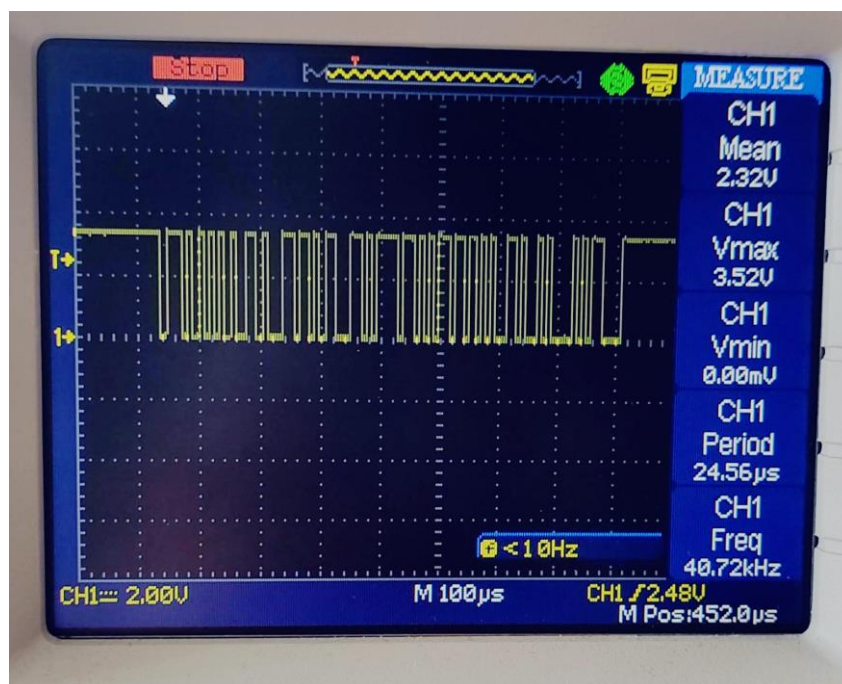


Figure 11

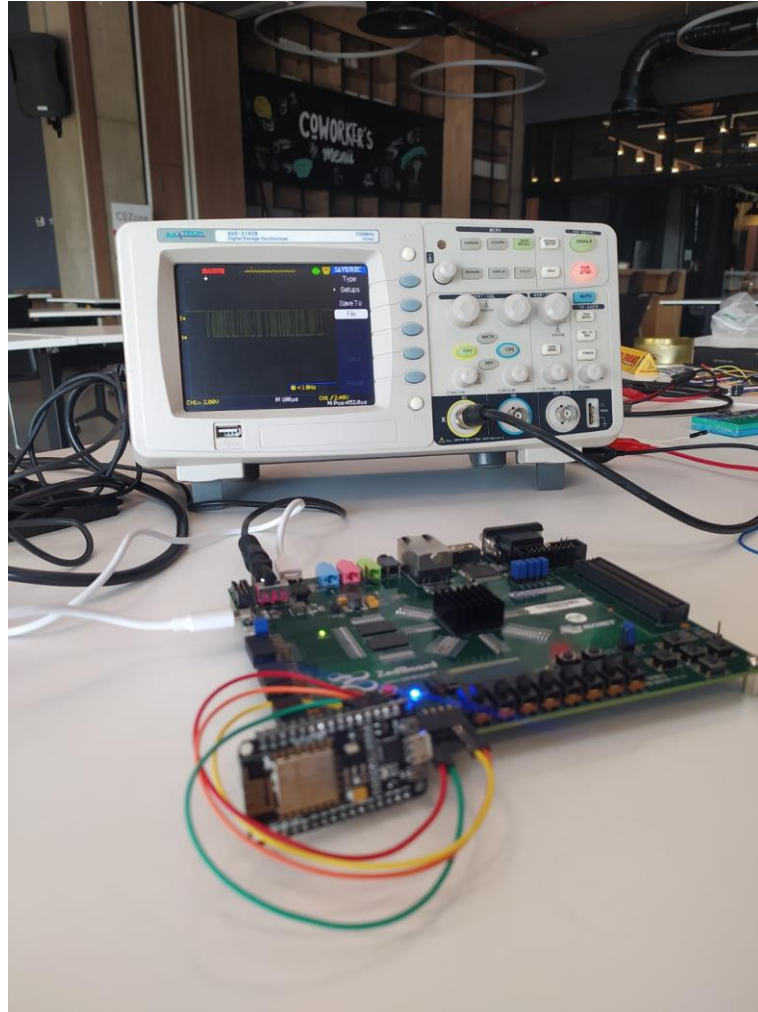


Figure 12

6. Results

Development and Implementation of a custom Linux OS on ZedBoard

Several actions, files and tools have been required to be able to develop and implement a specific OS into a specific board, in this case ZedBoard.

On one hand, these required files are responsible for all the tasks that ZedBoard needs to execute in a particular order to boot Linux OS on it. These tasks range from prepared and configure the PL and PS in the first stages to boot the operating system itself at last instance ultimately.

On the other hand, the entire set of necessary actions which have been necessary to configure and build these files throughout this thesis will be remembered and briefly summarized below. They will be also accompanied by the most relevant information and tools which should be known about them.

The Boot Image (“boot.bin”), which initializes the processor resources (FSBL file), configure the programmable logic (BitStream file), and loads the Linux kernel (U-Boot file).

The Device Tree Binary File (“devicetree.dtb”), which is obtained from the Device Tree Source File and loaded into the DDR memory by U-Boot. It allows the kernel to know every detail about the hardware in which it is working on.

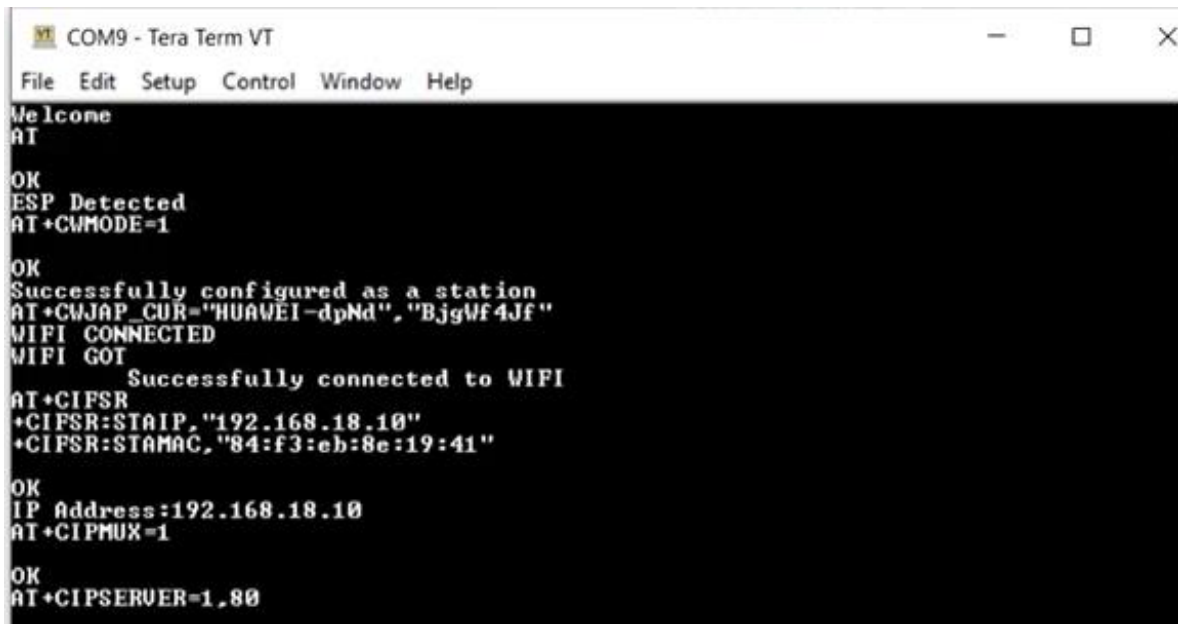
The Linux Kernel File (“uImage”), which is also loaded into the DDR memory by U-Boot. It initializes the system hardware and mounts the root file system.

The Root File System Image (“uramdisk.image.gz” or another partition with the OS files and folders), which can be a small custom image which is loaded into the DDR memory or can be a complete pre-configured OS which contains all necessary files and libraries to work as a normal PC. [9]



Figure 13

We observed the data transfer between the Zedboard and the WIFI module ESP 8266 using the Tera Term application. These parts are observed in **figures 14,15,16,17**.



```
COM9 - Tera Term VT
File Edit Setup Control Window Help
Welcome
AT
OK
ESP Detected
AT+CWMODE=1
OK
Successfully configured as a station
AT+CWJAP_CUR="HUAWEI-dpNd","BjgWf4Jf"
WIFI CONNECTED
WIFI GOT IP
Successfully connected to WIFI
AT+CIFSR
+CIFSR:STAIP,"192.168.18.10"
+CIFSR:STAMAC,"84:f3:eb:8e:19:41"
OK
IP Address:192.168.18.10
AT+CIPMUX=1
OK
AT+CIPSERVER=1,80
```

Figure 14

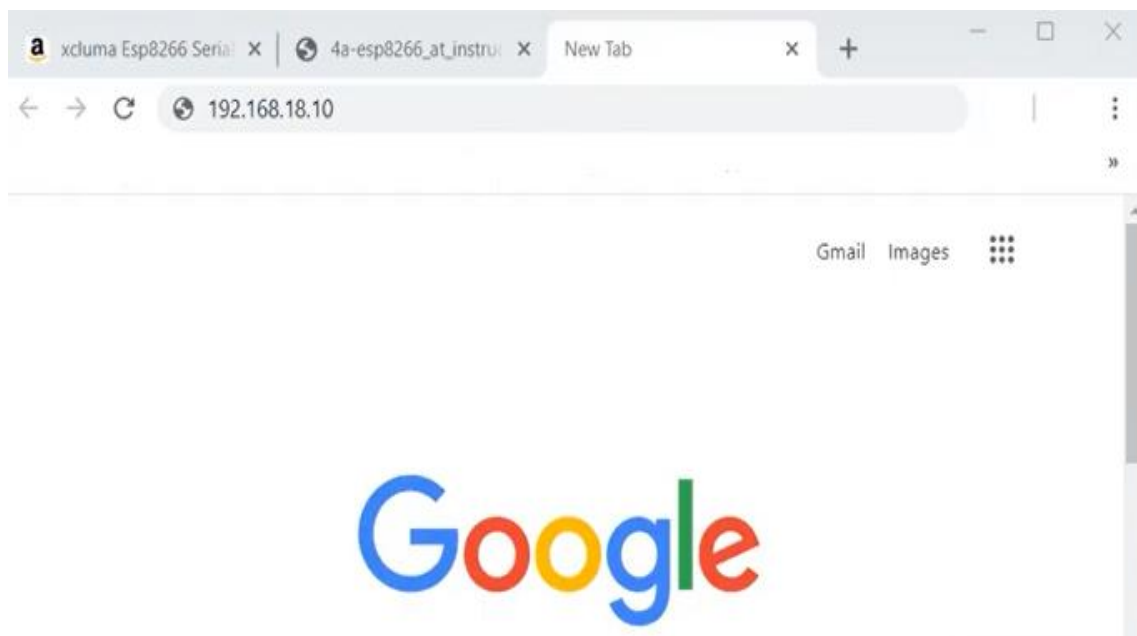
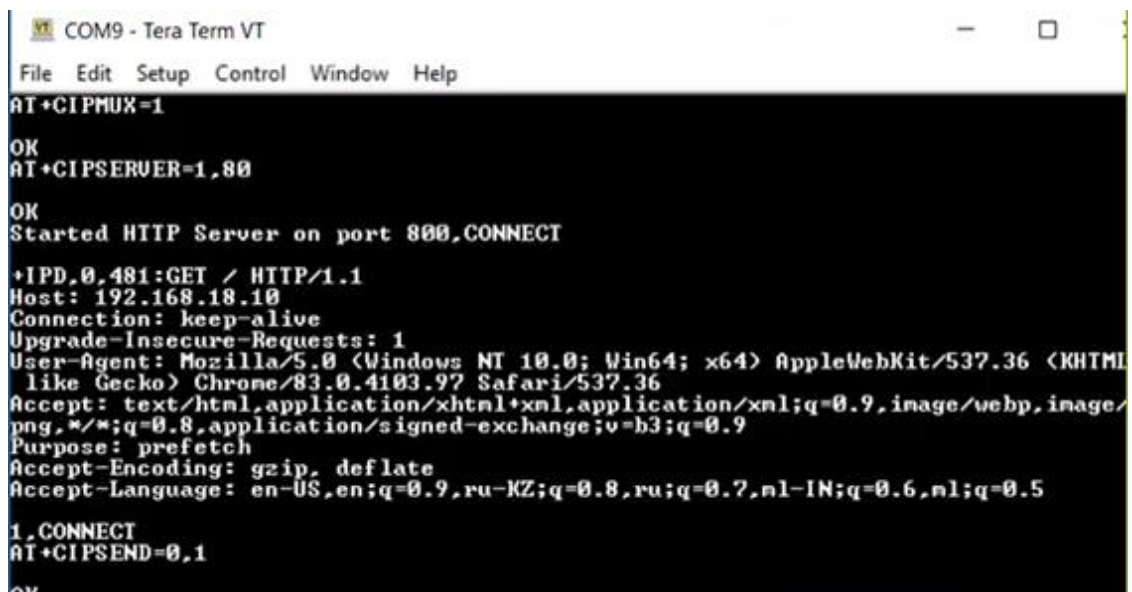


Figure 15



```
COM9 - Tera Term VT
File Edit Setup Control Window Help
AT+CIPMUX=1
OK
AT+CIPSERVER=1,80
OK
Started HTTP Server on port 800,CONNECT
+IPD,0,481:GET / HTTP/1.1
Host: 192.168.18.10
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
like Gecko) Chrome/83.0.4103.97 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Purpose: prefetch
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru-KZ;q=0.8,ru;q=0.7,nl-IN;q=0.6,nl;q=0.5
1,CONNECT
AT+CIPSEND=0,1
OK
```

Figure 16

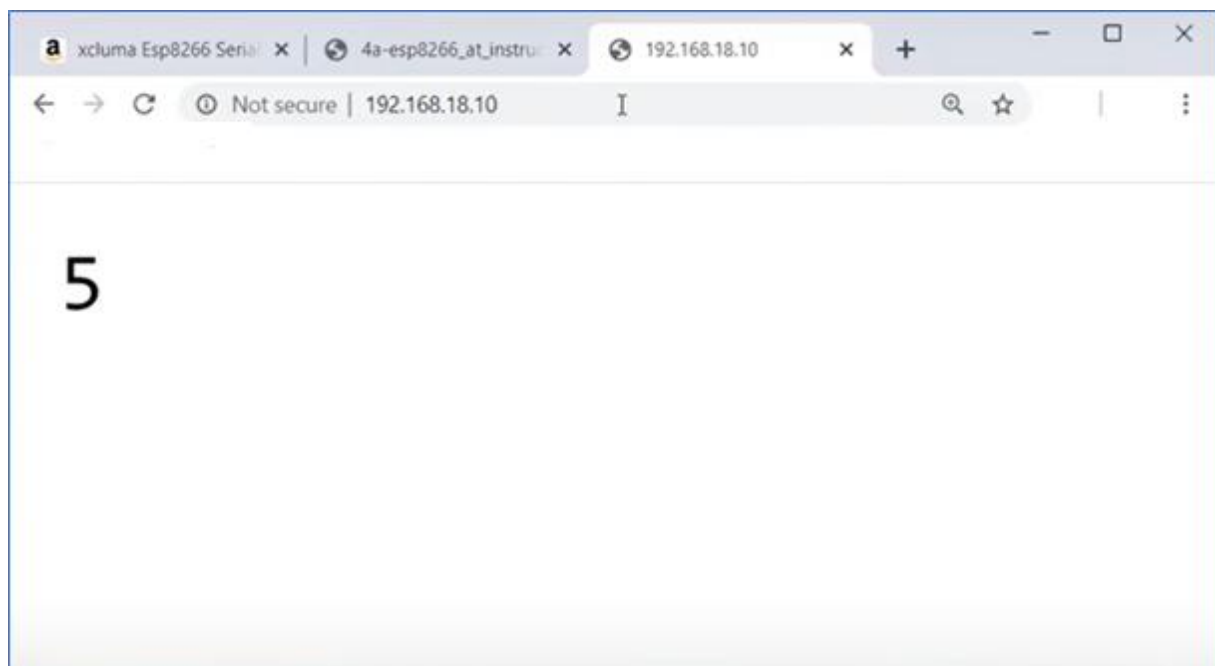


Figure 17

7. Discussions

Why on ZedBoard

There are a wide range of FPGAs, such as Artix, Kintex, Spartan, Virtex, Zynq ZC70X, etc. Nevertheless, ZedBoard is the chosen board to be used and programmed in this case. Therefore, the reasons to select ZedBoard instead of any of the previous boards are showed below and will be briefly explained:

Different memories types

Dual core ARM Cortex-A9

Great variety of peripherals

SoC architecture

Different Memories Types

ZedBoard includes several kinds of memories, such as a 512MB DDR3 memory, a 256MB flash memory and a SD slot. This gives the flexibility of allowing small-size systems to be stored in the flash memory, with the advantages that this kind of memory involved; and allowing huge-size systems to be stored in an external SD Card.

Therefore, a very fast or a heavy embedded Linux can be developed on ZedBoard.

Dual Core ARM Cortex-A9

ARM is present in most of current Smartphones (about 95% in 2010), and a wide range of smart TVs and laptops (35% and 10% respectively). Therefore, it is the perfect processor for the board.

In addition, ZedBoard not only includes a simple core ARM, but also includes a dual core ARM Cortex-A9. Therefore, the processor will not be a bottleneck for the developed applications on the board in any case.

Great Variety of Peripherals

The ZedBoard provides a wide range of interfaces to connect the most common peripherals and devices, such as monitors, keyboards, speakers, internet connection, etc. The most important interfaces which the ZedBoard provides are the followings:

Audio line-in, line-out, headphone and microphone.

Ethernet.

HDMI and VGA.

OLED display.

SD Card.

USB.

Therefore, the embedded Linux version will be able to use a monitor, a mouse and a keyboard; and will be able also to have internet connection.

SoC Architecture

The ZedBoard is an evaluation and development board based on the Xilinx Zynq-7000 All Programmable SoC (System-on-a-chip). This board allows creating a Linux, Android, Windows or other OS/RTOS-based design.

Therefore, the ZedBoard is not only a FPGA, but it is a Programmable SoC device. But, what is the difference between a FPGA and a programmable SoC device? The main difference is that the SoC combines the processing system (PS) with the programmable logic (PL); as a result, it has a higher speed and a less size than a traditional FPGA.

In particular, the ZedBoard combines a dual Corex-A9 Processing System (PS) with 85,000 Series-7 Programmable Logic (PL) cells. [10]

8. Conclusion

The building of the Embedded Linux with Linuxlink has advantages and disadvantages:

The main advantage is the simplicity for building the complete system with the online version, because the menus are classified into different matters like Kernel, RFS, packages, etc. The Factory version has several options which are different from the online options. Linuxlink checks for problems in the chosen configuration and warns the designer offering him a solution as well.

The main disadvantage is the poor documentation available. Besides, documents are divided and spreaded so it is difficult to find and follow them.

The Vivado software helps the designer with automated processes and predefined configurations. It is difficult to understand all the options included in Vivado but the “speedways” assists in using it without having to deal with all these options and allowing user to focus on his target.

The SDK software has several templates with illustrative names which make easy to guess the purpose of each one. [11]

After performing these operating system in the first part, we continued second part with providing communication between Zedboard and ESP8266 Wifi module.

9. Acknowledgements

We would like to thank our supervisor Orhan GAZI, for his guidance and help throughout the project. We further like to thank electronic and communication engineer Fatih GENC for the preparation of experimental setup.

10. References

- [1] : Article Reference: Article Reference: Ginés Hidalgo Martínez, Implementing an Embedded Linux System in Xilinx Zynq, 2013/2014
- [2] : Article Reference: Miguel del Castillo Campos, Embedded Linux Integration Using Linuxlink Tool and Peripheral Driver Development for Zedboard, Departamento de Ingeniería Telemática y Electrónica, 2008
- [3] : Website Reference: <https://www.circuitbasics.com/basics-uart-communication/>
- [4] : Article Reference: Miguel del Castillo Campos, Embedded Linux Integration Using Linuxlink Tool and Peripheral Driver Development for Zedboard, Departamento de Ingeniería Telemática y Electrónica, 2008
- [5] : Article Reference: Article Reference: Ginés Hidalgo Martínez, Implementing an Embedded Linux System in Xilinx Zynq, 2013/2014
- [6] : Article Reference: Miguel del Castillo Campos, Embedded Linux Integration Using Linuxlink Tool and Peripheral Driver Development for Zedboard, Departamento de Ingeniería Telemática y Electrónica, 2008
- [7] : Website Reference: https://en.wikipedia.org/wiki/Tera_Term
- [8] : Article Reference: Miguel del Castillo Campos, Embedded Linux Integration Using Linuxlink Tool and Peripheral Driver Development for Zedboard, Departamento de Ingeniería Telemática y Electrónica, 2008
- [9] : Article Reference: Article Reference: Ginés Hidalgo Martínez, Implementing an Embedded Linux System in Xilinx Zynq, 2013/2014
- [10] : Article Reference: Article Reference: Ginés Hidalgo Martínez, Implementing an Embedded Linux System in Xilinx Zynq, 2013/2014
- [11] : Article Reference: Miguel del Castillo Campos, Embedded Linux Integration Using Linuxlink Tool and Peripheral Driver Development for Zedboard, Departamento de Ingeniería Telemática y Electrónica, 2008