Wireless Communication Between Embedded Systems Based on FPGA

USER MANUAL

By

Daryl George

Department of Electronics and Telecommunication Engineering

Fr. Conceicao Rodrigues Institute of Technology, Vashi

Table of Contents

List of figures

Chapter 1

1 INTRODUCTION

This user manual details the steps and design flow of running the project entitled "Wireless Communication Between Embedded Systems Based on FPGA".
The functional block diagram of this project is as given below in Figure 1.

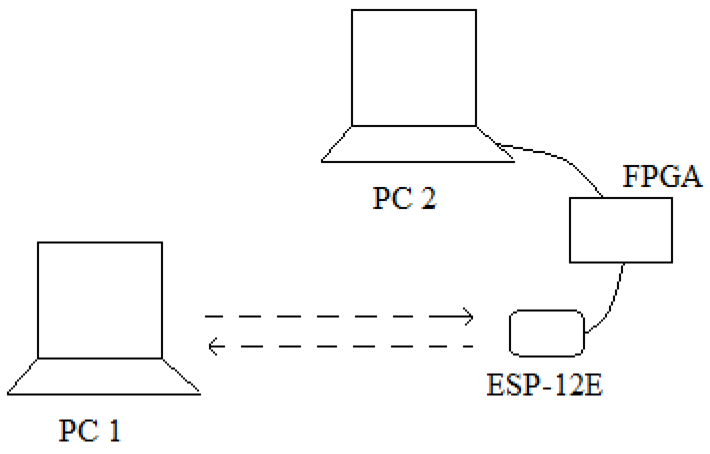


Figure 1.1 Functional Block Diagram

With this project the client side can send data to the server side which has a Wi-Fi module and FPGA connected to it.
This data can be stored in the FPGA and can be later read by the client by using the write and read operation explained in the later chapters.
The FPGA board used for this project is the Nexys 3 FPGA board by Xilinx DIGILENT. The following figure shows the image of the FPGA board.

Features of Nexys 3 FPGA Board

- Xilinx Spartan-6 LX16 FPGA in a 324-pin BGA package
- 16Mbyte Cellular RAM (x16)
- 16Mbytes SPI (quad mode) PCM non-volatile memory
- 16Mbytes parallel PCM non-volatile memory
- 10/100 Ethernet PHY
- On-board USB2 port for programming & data transfer
- USB-UART and USB-HID port (for mouse/keyboard)
- 8-bit VGA port

- 100MHz CMOS oscillator
- GPIO includes 8 LEDs, 5 buttons, 8 slide switches and 4-digit seven-segment display.

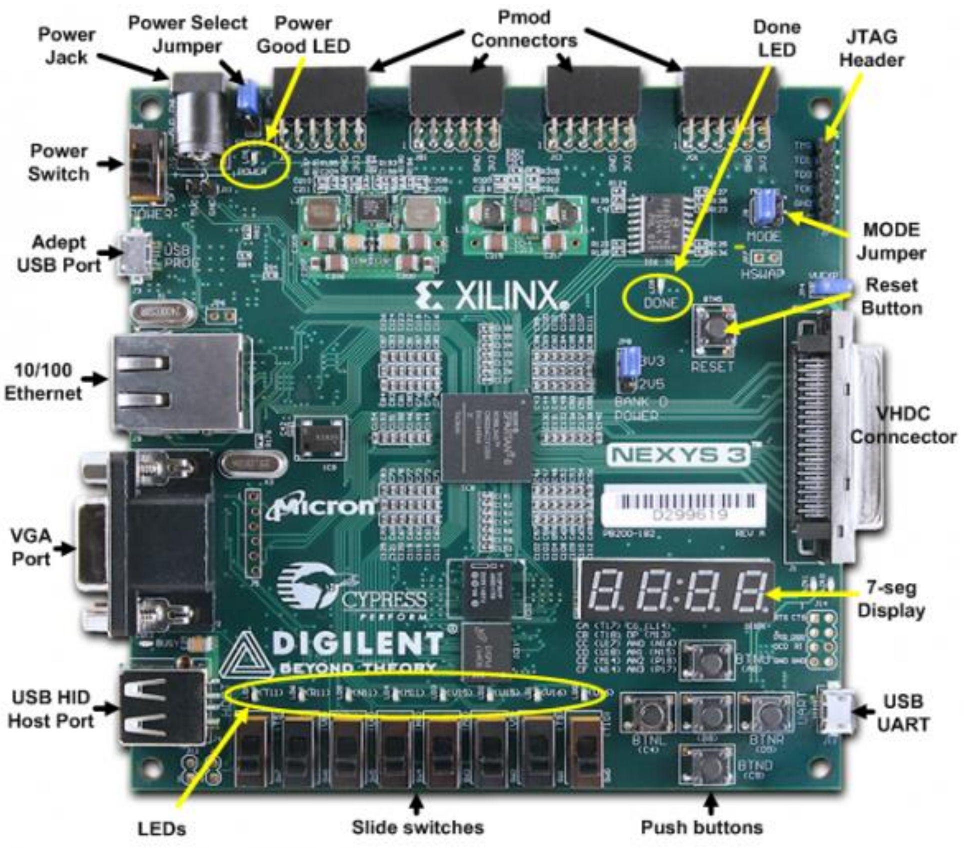


Figure 1.2 Nexys 3 FPGA board

And the Wi-Fi module used for this project is ESP8266 12 E (NodeMCU) module as shown in Figure 3.

Figure 1.3 NodeMCU Wi-Fi module

Features of ESP8266-12E (NodeMCU)

- 64 KiB of instruction RAM, 96 KiB of data RAM.
- IEEE 802.11 b/g/n Wi-Fi.
- Integrated TR switch, LNA, power amplifier and matching network.
- WEP or WPA/WPA2 authentication, or open networks.
- 16 GPIO pins.

Before starting the configurations please note that the client and server PC are running on the same Wi-Fi network.

All the codes and datasheets needed to implement this project are provided in the GitHub repository.

The link for the repository is : https://github.com/darylgeorge/FPGA_NodeMCU

Chapter 2

2 CONFIGURATIONS

In this project we have to configure three parts separately in order to successfully implement this project.

- Client side PC
- NodeMCU Module
- FPGA Board

Let us now configure each module individually. Follow the steps and the codes and datasheets are available in the repository.

2.1 Code for Server-side NodeMCU module

The server side module selected for this project is NodeMCU module that is ESP8266 12-E module from Expressif systems.
It's a module that works in the Wi-Fi domain which can be programmed via Arduino IDE (Integrated Development Environment).
In order to program the NodeMCU as server we have to follow some steps to make the Arduino IDE compatible for coding the Wi-Fi module as it is not listed in its default boards library.
Steps to make Arduino IDE compatible with NodeMCU module:

- Install Arduino IDE from its official website or copy paste the following link in the browser and download.
- Download Link : http://www.arduino.cc/en/main/software
- Once downloaded click on the downloaded setup file and follow the installation procedures.
- Once installed you will see the Arduino IDE icon on your desktop.
- Click on it to launch the IDE.

- Once the IDE is launched go to File and click on Preferences.

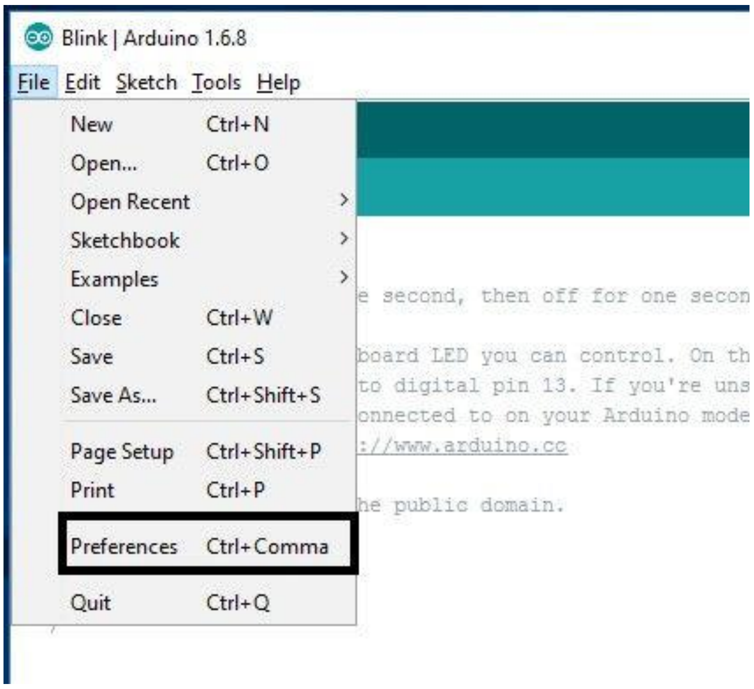
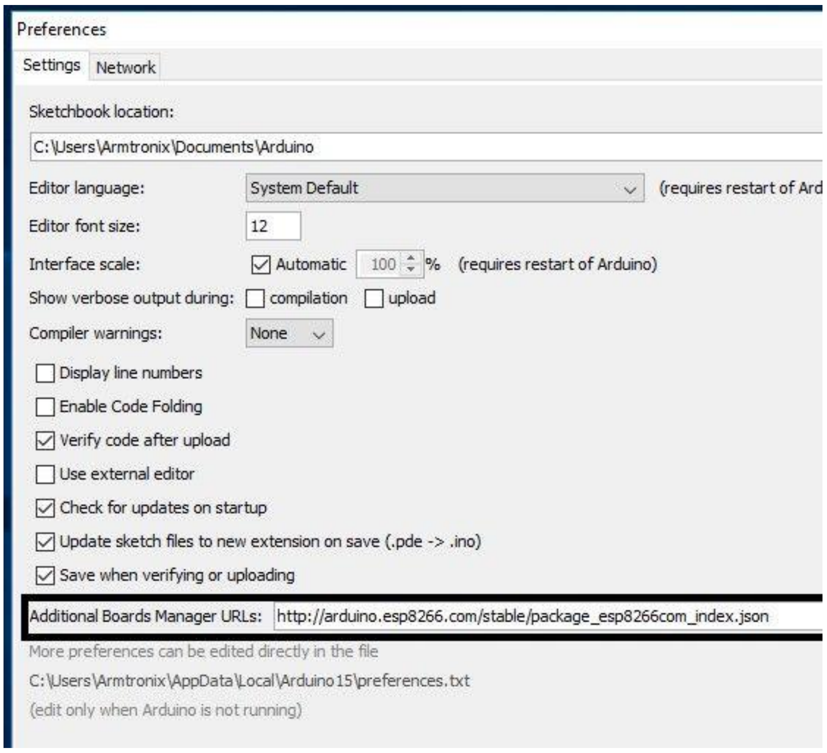


Figure 2.1 File>Preferences

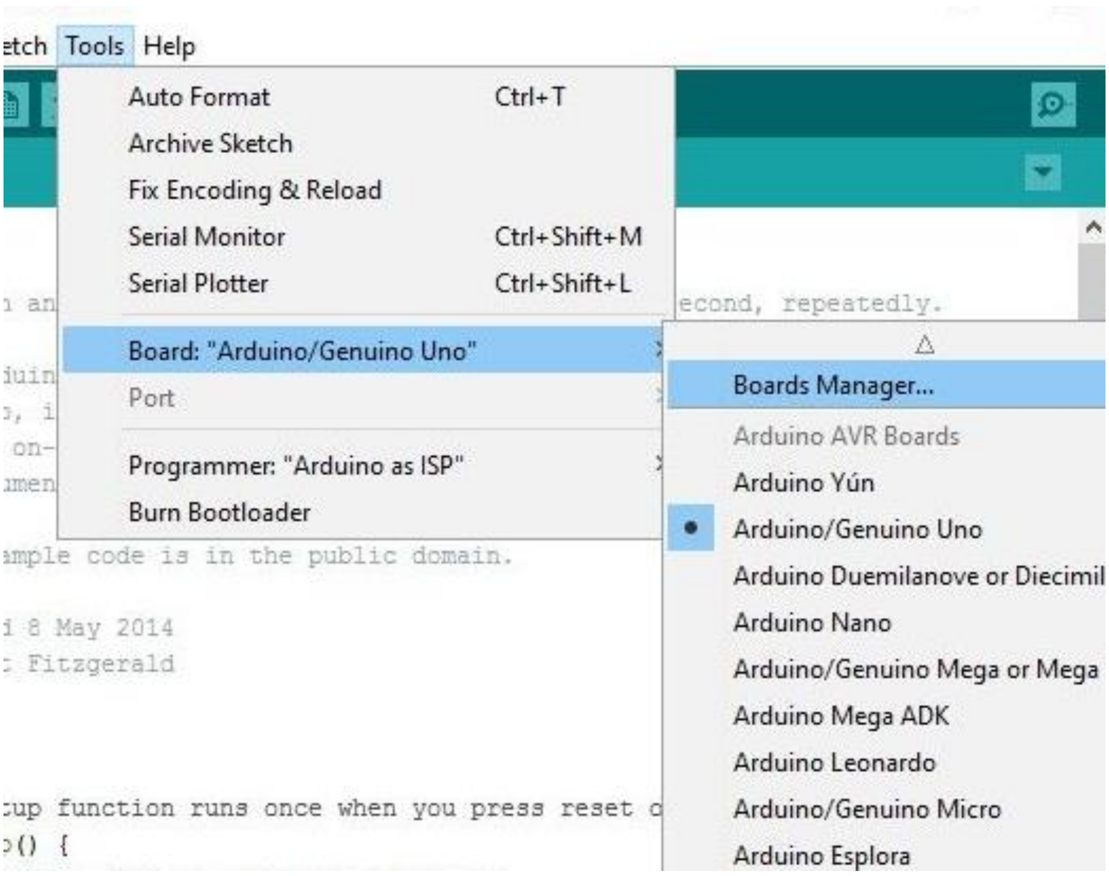
- Now we have to add the ESP8266 board manager.
- Go to “Additional Boards Manager URLS:” and type in the following url: “http://arduino.esp8266.com/stable/package_esp8266com_index.json”.



Sketchbook location:
C:\Users\Armtronix\Documents\Arduino
Editor language: System Default (requires restart of Ard
Editor font size: 12
Interface scale: [✓] Automatic 100 % (requires restart of Arduino)
Show verbose output during: [ ] compilation [ ] upload
Compiler warnings: None
[ ] Display line numbers
[ ] Enable Code Folding
[✓] Verify code after upload
[ ] Use external editor
[✓] Check for updates on startup
[✓] Update sketch files to new extension on save (.pde -> .ino)
[✓] Save when verifying or uploading
Additional Boards Manager URLs: http://arduino.esp8266.com/stable/package_esp8266com_index.json
More preferences can be edited directly in the file
C:\Users\Armtronix\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Figure 2.2 Boards Manager

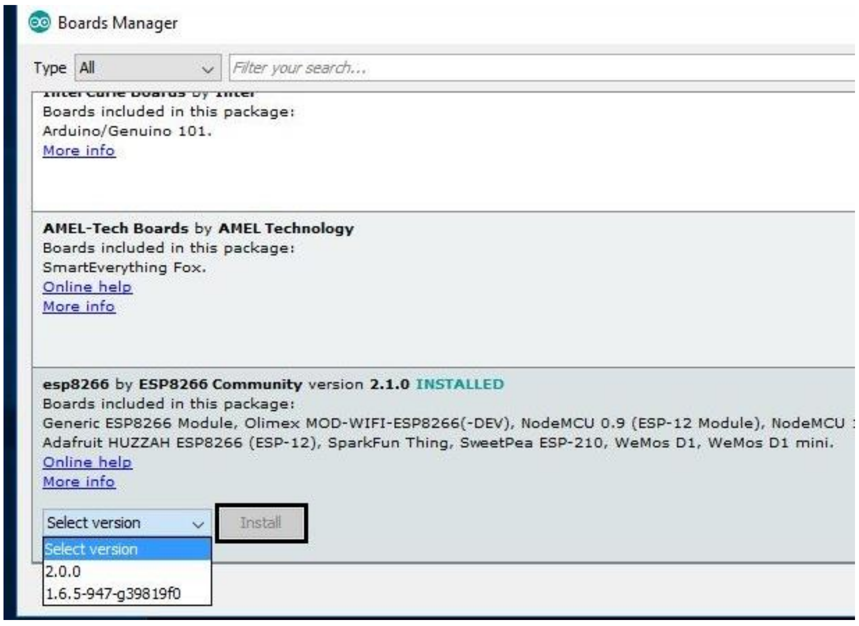
- Close the Preference dialogue box.
- Now open the tools in that select Board: “Arduino/Genuino Uno” and click on the Boards Manager as shown in the figure.



```
a an                                      econd, repeatedly.
uin
), i
 on-
umen
ample code is in the public domain.

i 8 May 2014
: Fitzgerald


:up function runs once when you press reset o
p() {
```

Figure 2.3 Tools>Boards Manager

- The Boards Manager window opens, scroll the window page to bottom till you see the module with the name ESP8266. Once we get it, select that module and select version and click on the Install button.
- When it is installed it shows Installed in the module as shown in the figure and then close the window.



Boards Manager

Type All | Filter your search...

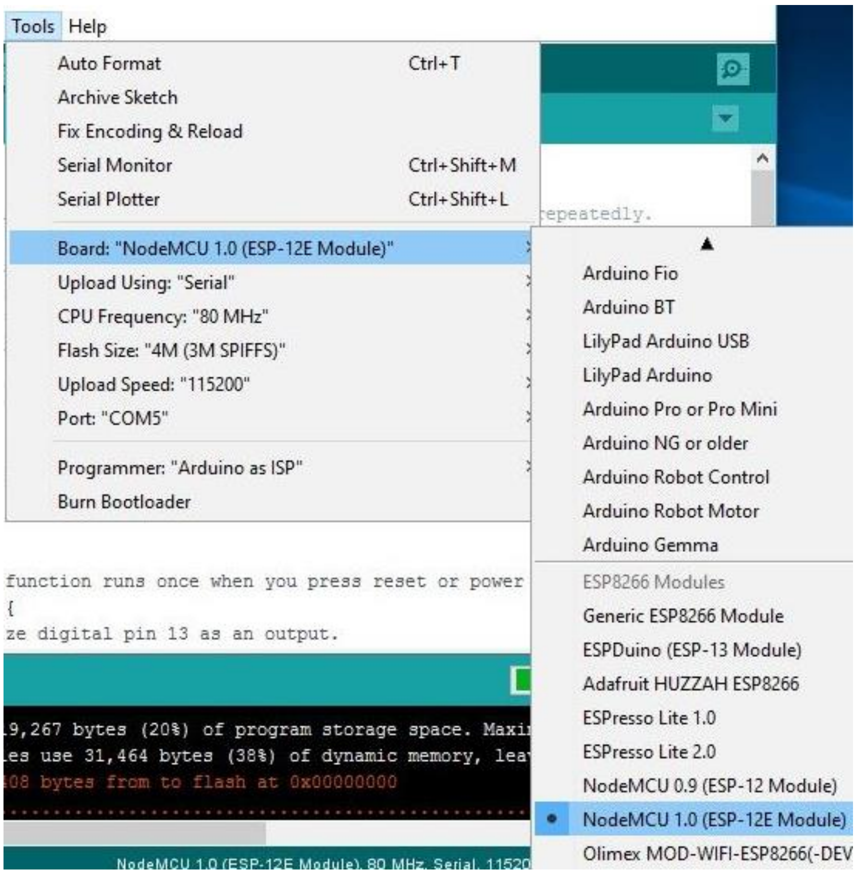
Intel Curie Boards by Intel
Boards included in this package:
Arduino/Genuino 101.
More info

AMEL-Tech Boards by AMEL Technology
Boards included in this package:
SmartEverything Fox.
Online help
More info

esp8266 by ESP8266 Community version 2.1.0 INSTALLED
Boards included in this package:
Generic ESP8266 Module, Olimex MOD-WIFI-ESP8266(-DEV), NodeMCU 0.9 (ESP-12 Module), NodeMCU
Adafruit HUZZAH ESP8266 (ESP-12), SparkFun Thing, SweetPea ESP-210, WeMos D1, WeMos D1 mini.
Online help
More info

Figure 2.4 ESP8266 version install

- To run the esp8266 with Arduino we have to select the Board: “Arduino/Genuino Uno” and then change it to NodeMCU 1.0 (ESP-12E Module) or other esp8266 modules depending on what you have.



Board: "NodeMCU 1.0 (ESP-12E Module)"
Upload Using: "Serial"
CPU Frequency: "80 MHz"
Flash Size: "4M (3M SPIFFS)"
Upload Speed: "115200"
Port: "COM5"
Programmer: "Arduino as ISP"

```
repeatedly.
function runs once when you press reset or power
{
ze digital pin 13 as an output.
```

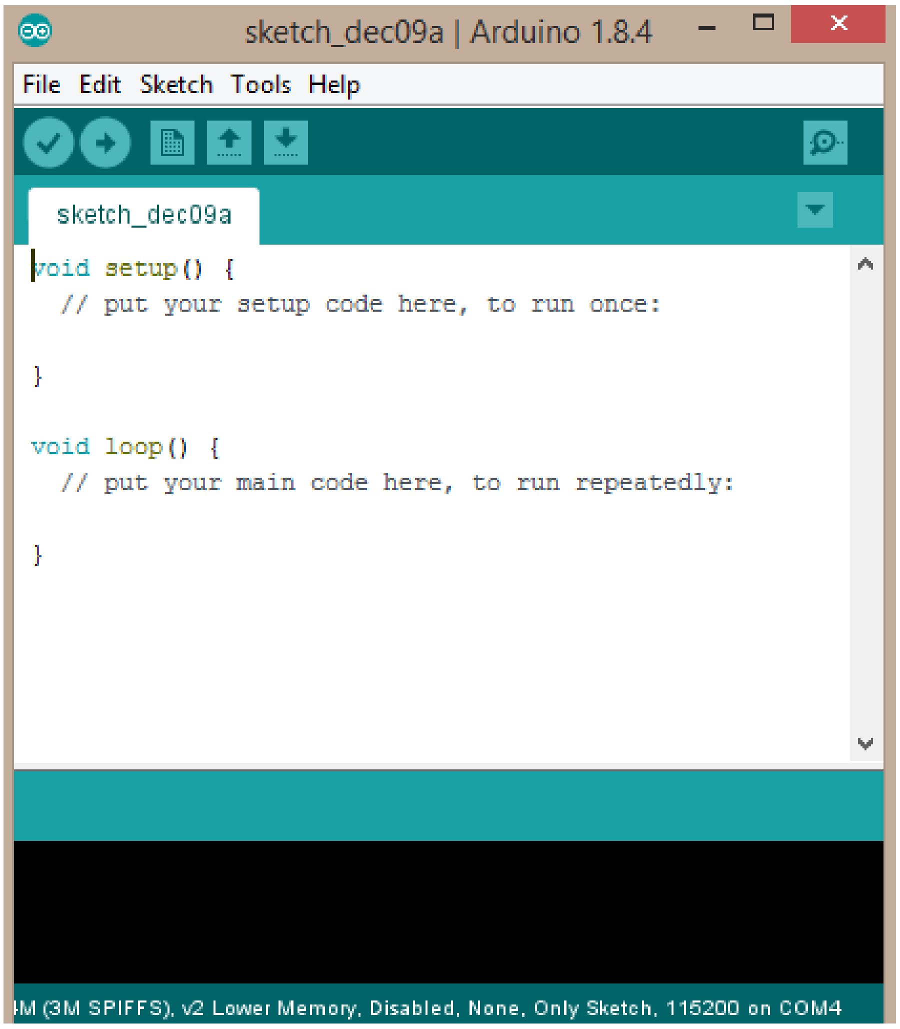
```
9,267 bytes (20%) of program storage space. Maxi
es use 31,464 bytes (38%) of dynamic memory, lea
08 bytes from to flash at 0x00000000
.....................................................
```

NodeMCU 1.0 (ESP-12E Module), 80 MHz, Serial, 11520

Figure 2.5 Selecting NodeMCU board

Steps to upload the code to NodeMCU:

- Go to File and select New.
- A new sketch will be launched where we can write the code.

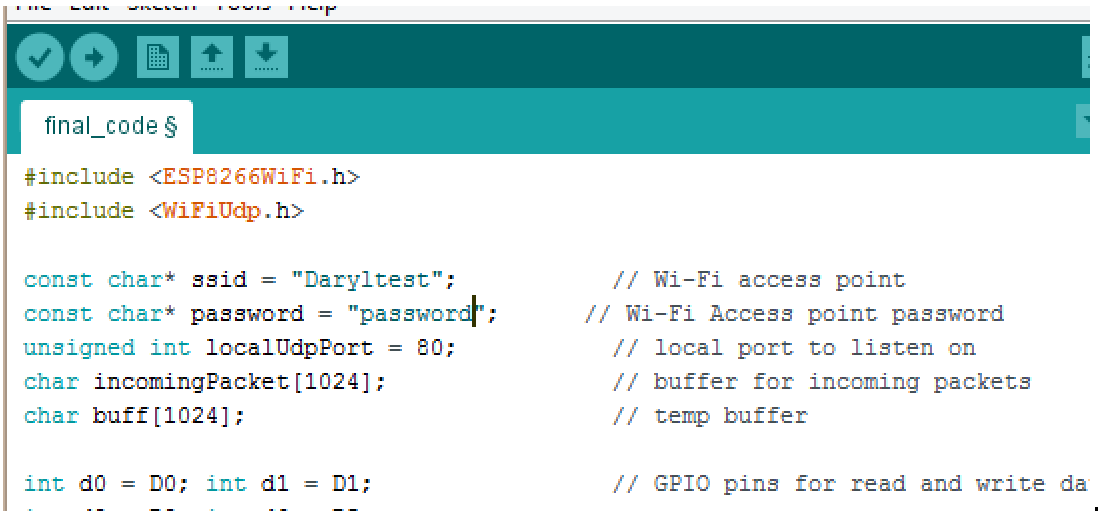


```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
M (3M SPIFFS), v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4
```

Figure 2.6 New Sketch

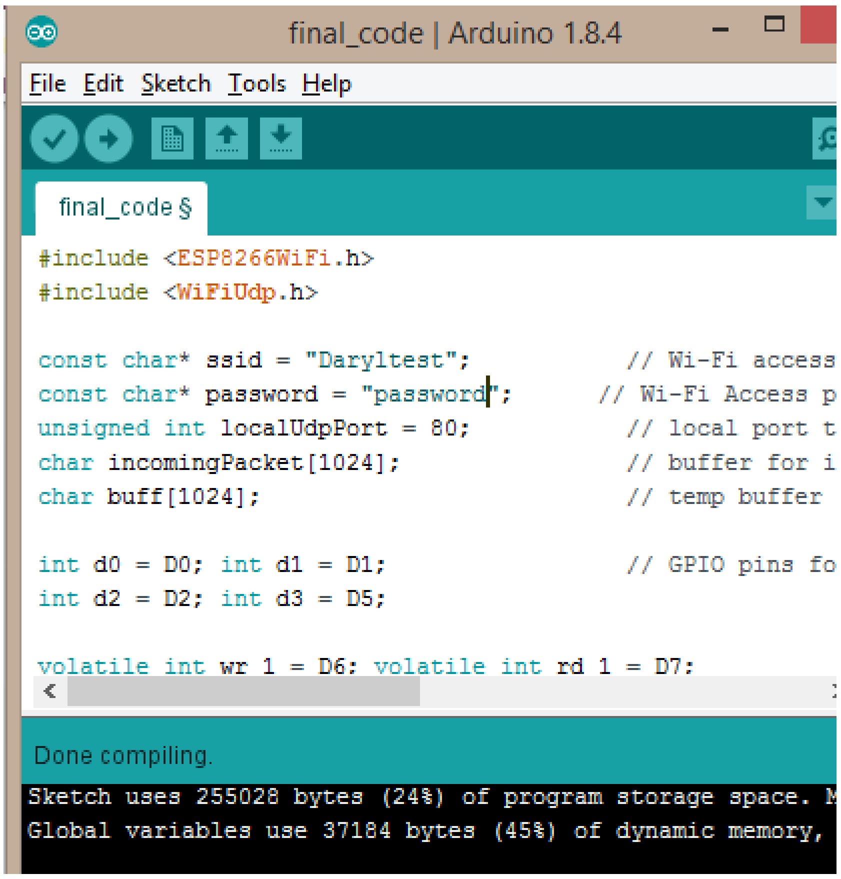
- The code for server side is also given in the repository.
- Go to FPGA_NodeMCU/Codes/NodeMCU code -server/
- The code is titled as “NodeMCU(server) code”.
- Copy this code to the newly opened sketch.
- In the code type in the Wi-Fi access point’s name and password in the ssid and pwd area as respectively as per the Wi-Fi network you are connected to



```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>

const char* ssid = "Daryltest";          // Wi-Fi access point
const char* password = "password";      // Wi-Fi Access point password
unsigned int localUdpPort = 80;          // local port to listen on
char incomingPacket[1024];               // buffer for incoming packets
char buff[1024];                         // temp buffer

int d0 = D0; int d1 = D1;                // GPIO pins for read and write da
```

Figure 2.7 Insert ssid and password

- Now click on compile icon (Icon with tick mark sign) or press “control+R” to compile the code.
- Once the code is compiled “Compiling done” message will show up in the black console window below.

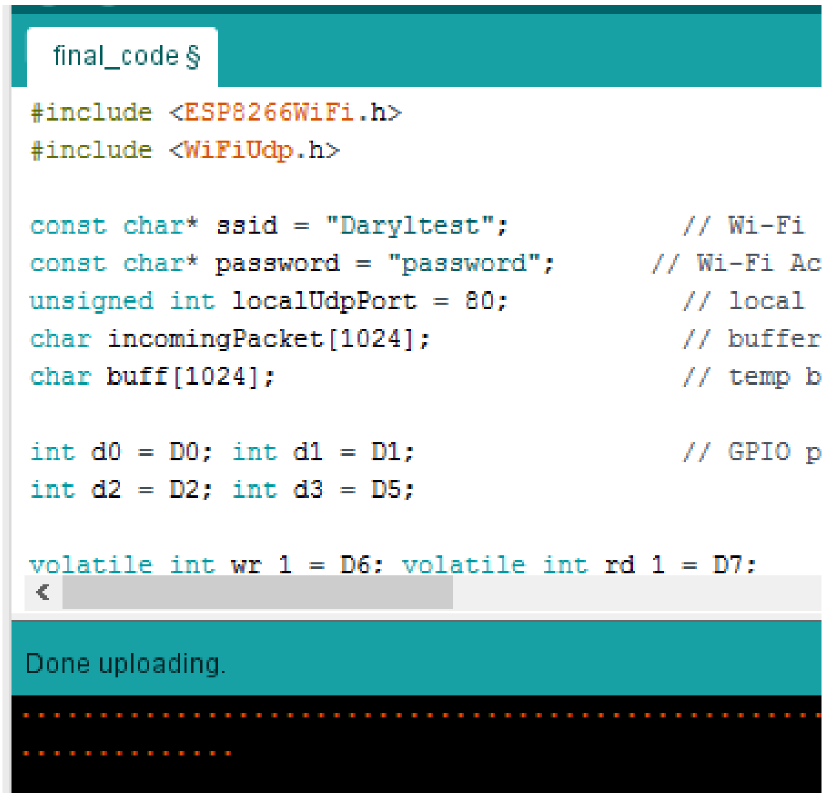


```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>

const char* ssid = "Daryltest";          // Wi-Fi access
const char* password = "password";      // Wi-Fi Access p
unsigned int localUdpPort = 80;          // local port t
char incomingPacket[1024];               // buffer for i
char buff[1024];                         // temp buffer

int d0 = D0; int d1 = D1;                // GPIO pins fo
int d2 = D2; int d3 = D5;

volatile int wr 1 = D6; volatile int rd 1 = D7;
```

```
Done compiling.
Sketch uses 255028 bytes (24%) of program storage space. M
Global variables use 37184 bytes (45%) of dynamic memory,
```

Figure 2.8 Compiling code

- Connect the NodeMCU module to the PC via USB port using USB-MicroUSB cable.
- Now go to Tools and select the COMM port that shows up where the module is connected.
- Now click on “upload” button (right arrow sign) or press “control+U” to upload the code to the module.
- Once if uploaded successfully it will Uploading Done” message will show up in the black console window below.



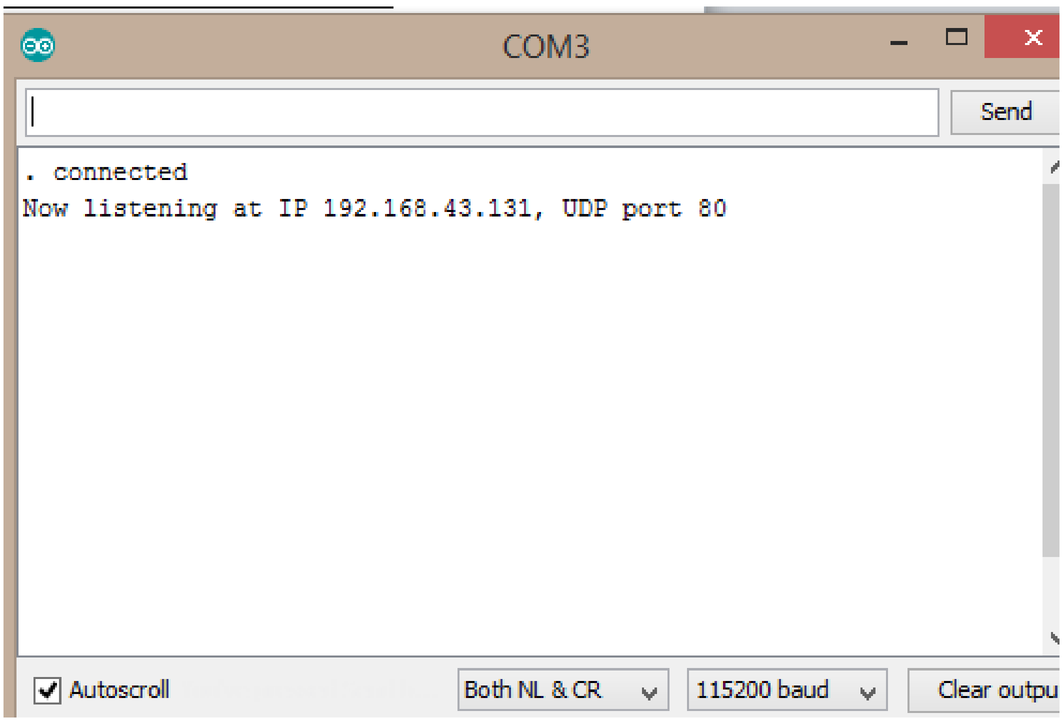
```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>

const char* ssid = "Daryltest";            // Wi-Fi
const char* password = "password";        // Wi-Fi Ac
unsigned int localUdpPort = 80;            // local
char incomingPacket[1024];                 // buffer
char buff[1024];                           // temp b

int d0 = D0; int d1 = D1;                  // GPIO p
int d2 = D2; int d3 = D5;

volatile int wr 1 = D6; volatile int rd 1 = D7;
```

```
Done uploading.
.....................................................
..............
```

Figure 2.9 Uploading the Code

- Now Go to Tools and select Serial Monitor or press “control+M” to open the Arduino serial monitor window.
- If code is uploaded without errors, the module will start searching for the Wi-Fi network mentioned in the above steps and try to establish its connection.
- Once connection is established it will show its local IP address and local Port Number provided by the Wi-Fi Network.

- Note these two parameters as they will be used for coding the client side program.



```
. connected
Now listening at IP 192.168.43.131, UDP port 80
```

Figure 2.10 Serial Monitor

2.2 Writing Client-side code for LINUX

The client side is code is programmed using the C language in LINUX terminal.
The above mentioned code is a Socket programming code in C language for UDP packet transmission.
The code for the above is given in the repository folder.

Steps to Program the client side terminal.

- Go to the repository folder and select FPGA_NodeMCU/Codes/PC code - client/Linux_code.
- The client side is code is named Linux_code.
- Download or copy the code in a text file.

- Now open the LINUX terminal by right-clicking and selecting new terminal.

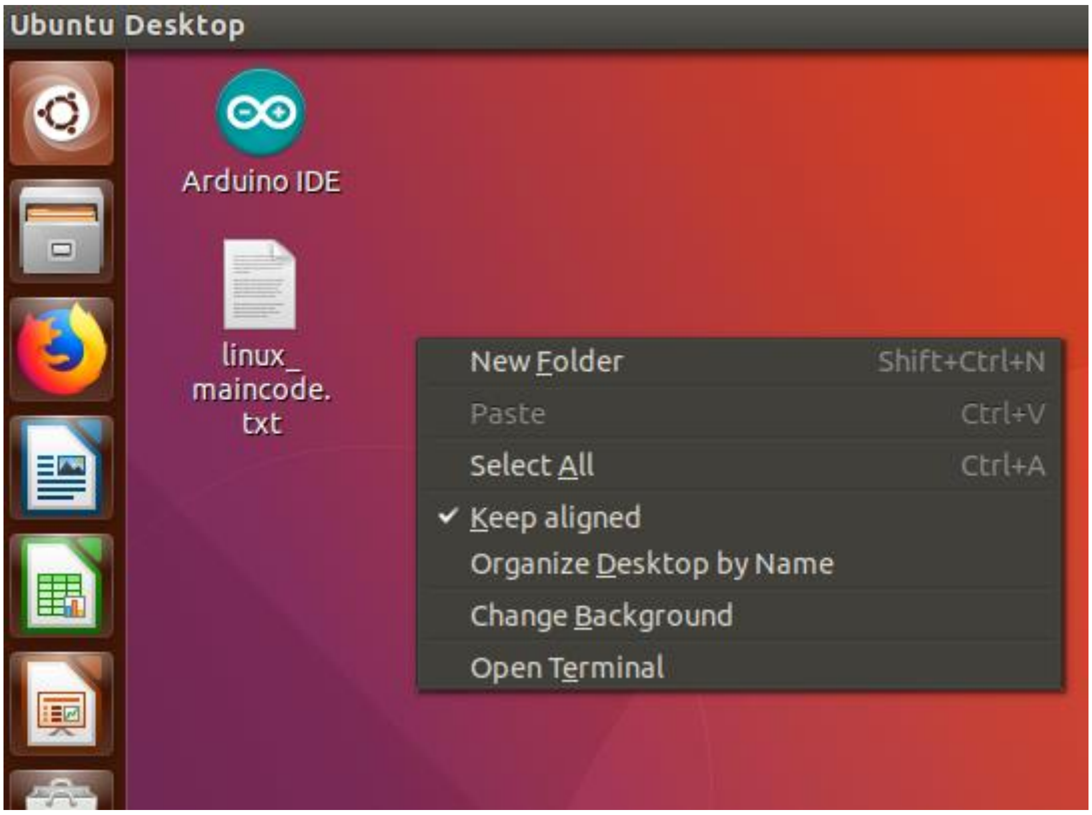
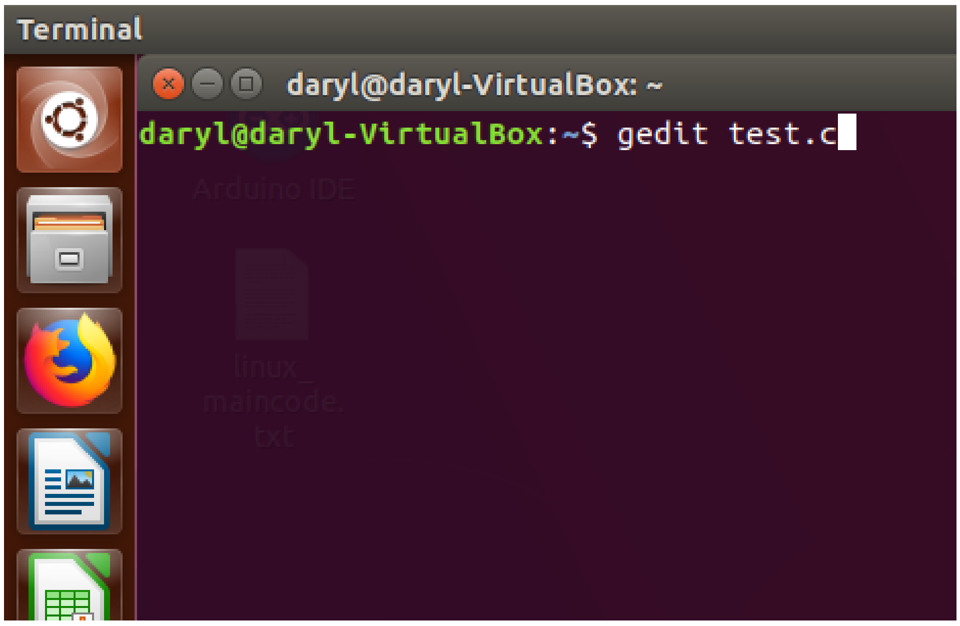


Figure 2.11 Open Linux terminal

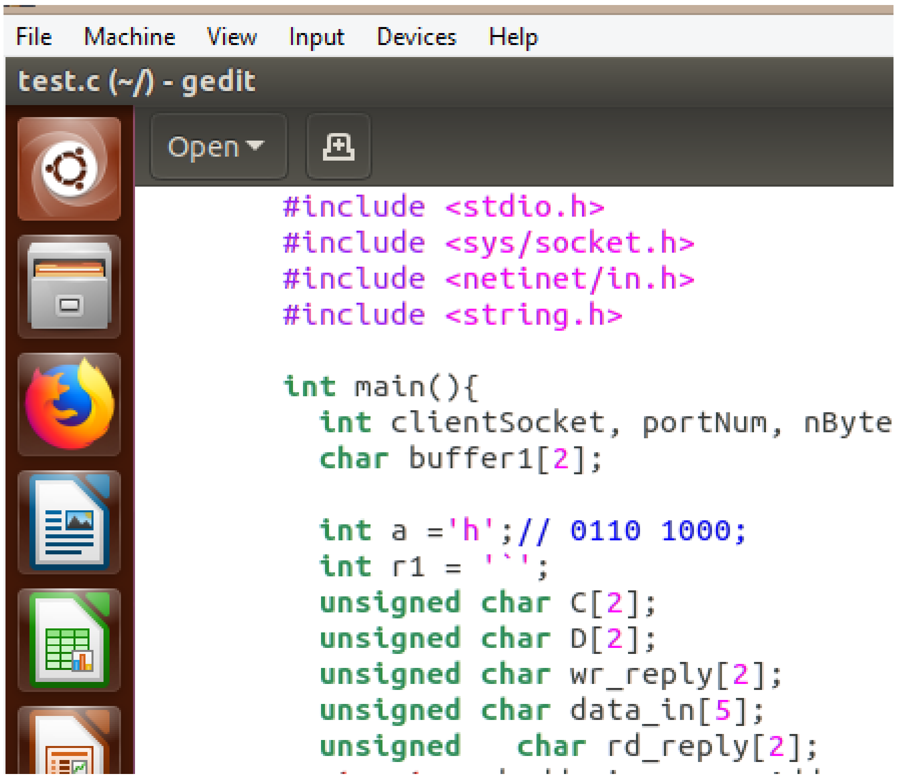
- Once the terminal window is opened, type : gedit test.c



```
daryl@daryl-VirtualBox:~$ gedit test.c
```

Figure 2.12 Open text editor

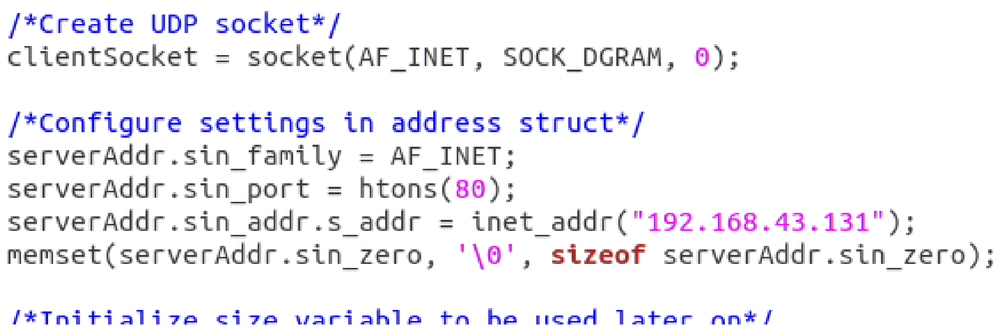
- This will open a text editor for C program titled “test”. You can change the name of the program code.
- Now copy the contents of “Linux_code” downloaded earlier.



```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <string.h>

int main(){
  int clientSocket, portNum, nByte
  char buffer1[2];

  int a ='h';// 0110 1000;
  int r1 = '`';
  unsigned char C[2];
  unsigned char D[2];
  unsigned char wr_reply[2];
  unsigned char data_in[5];
  unsigned   char rd_reply[2];
```

Figure 2.13 Copy code into the editor

- In the code you have to change the IP address and Port Number according to the new number obtained in the NodeMCU section and “save” the file.

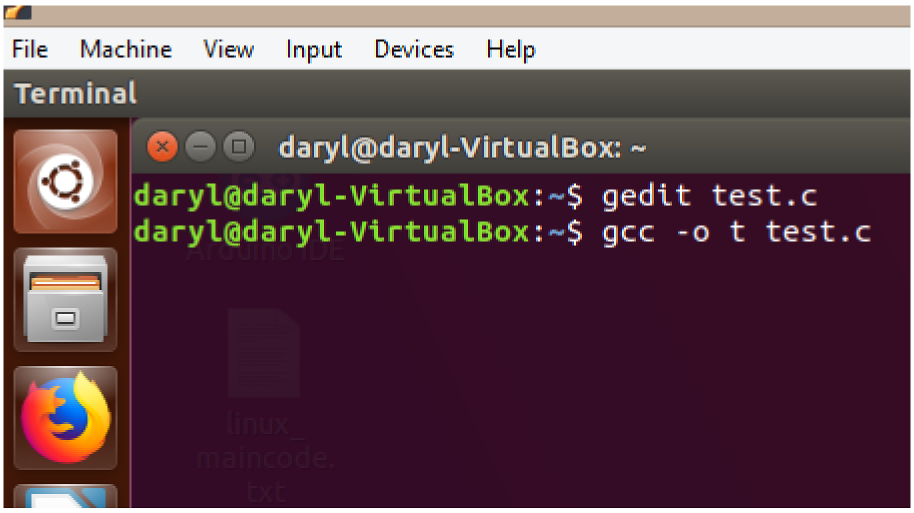


```
/*Create UDP socket*/
clientSocket = socket(AF_INET, SOCK_DGRAM, 0);

/*Configure settings in address struct*/
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(80);
serverAddr.sin_addr.s_addr = inet_addr("192.168.43.131");
memset(serverAddr.sin_zero, '\0', sizeof serverAddr.sin_zero);

/*Initialize size variable to be used later on*/
```

Figure 2.14 Change IP address and port number

- Close the editor and go back to the terminal.

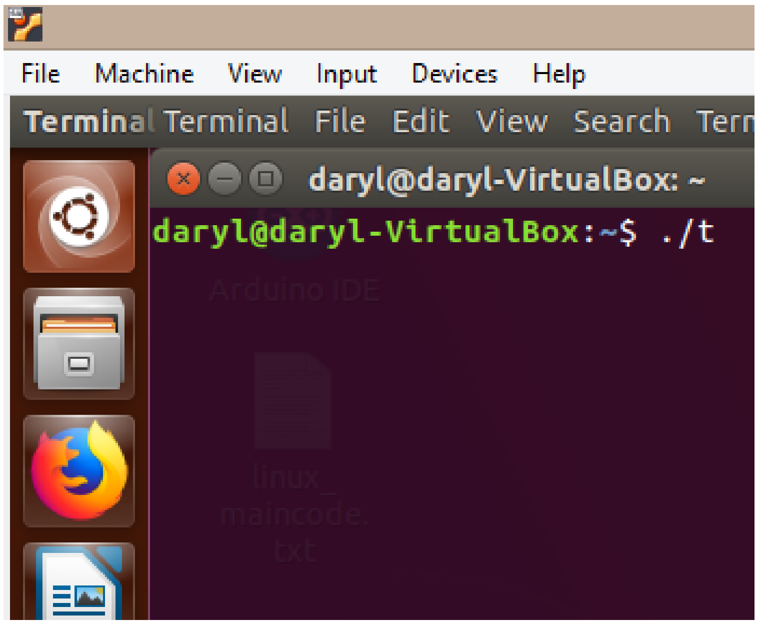
- Now type : gcc –o t test.c


File Machine View Input Devices Help
Terminal
daryl@daryl-VirtualBox: ~

```
daryl@daryl-VirtualBox:~$ gedit test.c
daryl@daryl-VirtualBox:~$ gcc -o t test.c
```

Figure 2.15 Compile code

- This will compile the code and show errors/warnings if any.
- Now to run the code type you will have to type: ./t


File Machine View Input Devices Help
Terminal Terminal File Edit View Search Term
daryl@daryl-VirtualBox: ~

```
daryl@daryl-VirtualBox:~$ ./t
```

Figure 2.16 Run the code

2.3 Configuring FPGA Board

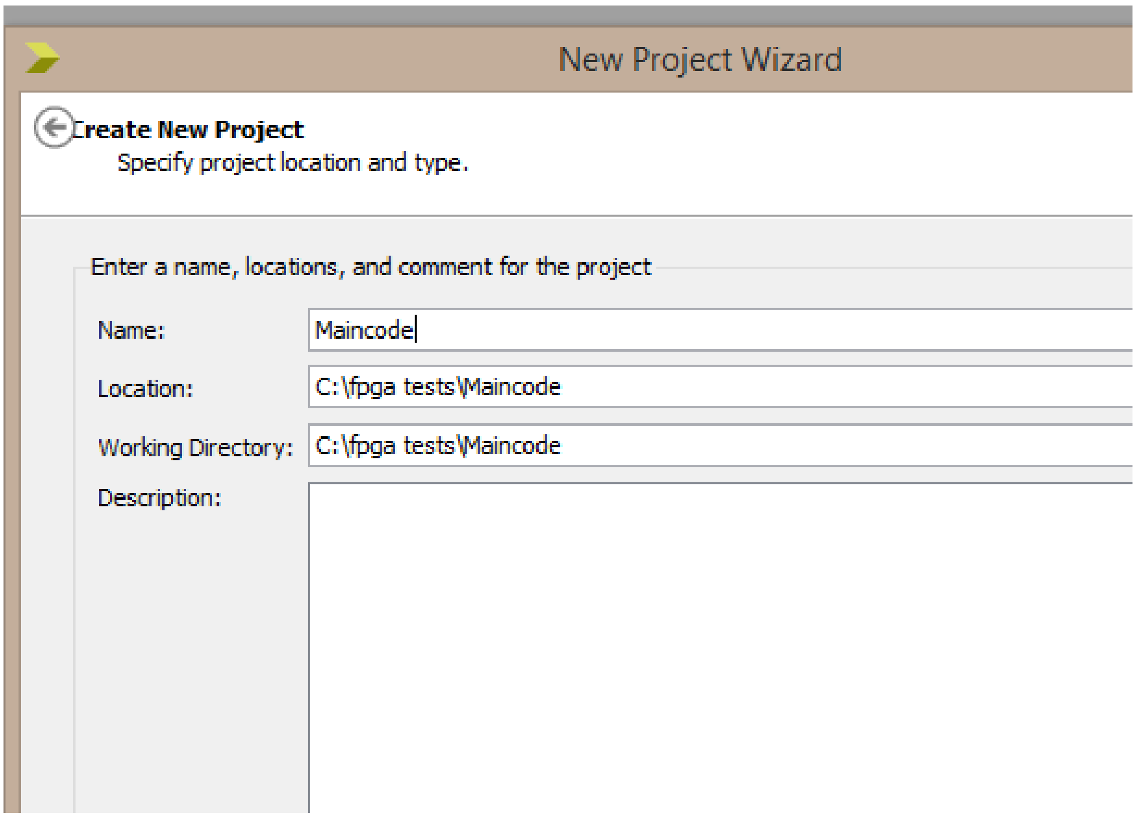
The FPGA board used for this project is Nexys 3 board from Xilinx DIGILENT which has Spartan 6 FPGA chip.

The code for FPGA board is written in Verilog HDL language using Xilinx ISE software. The software can be downloaded from the Xilinx official site. You can go to this link in order to download the software : https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools.html

Once the software has been installed follow the following steps.

Steps to upload Verilog code onto FPGA board.

- Launch Xilinx ISE and create "New Project".
- Select path of your working project directory and enter name of project click on Next.


New Project Wizard
Create New Project
Specify project location and type.
Enter a name, locations, and comment for the project

Name: Maincode

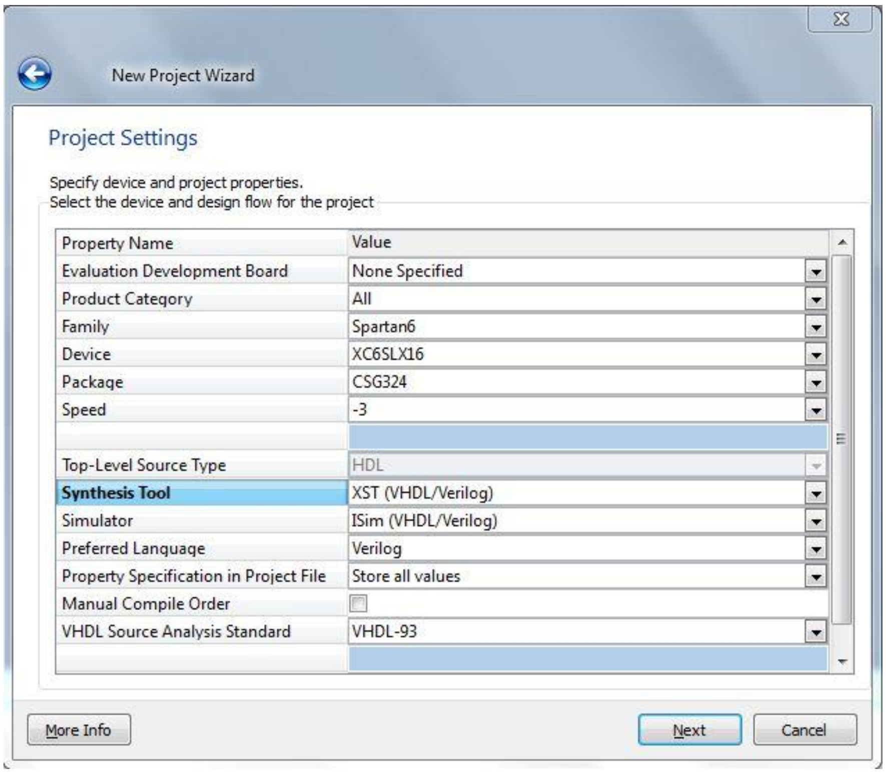
Location: C:\fpga tests\Maincode

Working Directory: C:\fpga tests\Maincode

Description:

Figure 2.17 New Project

- Select the Family, Device, Package and Speed of Xilinx board and also select your programming language(Verilog/VHDL). Here i am using Verilog language.



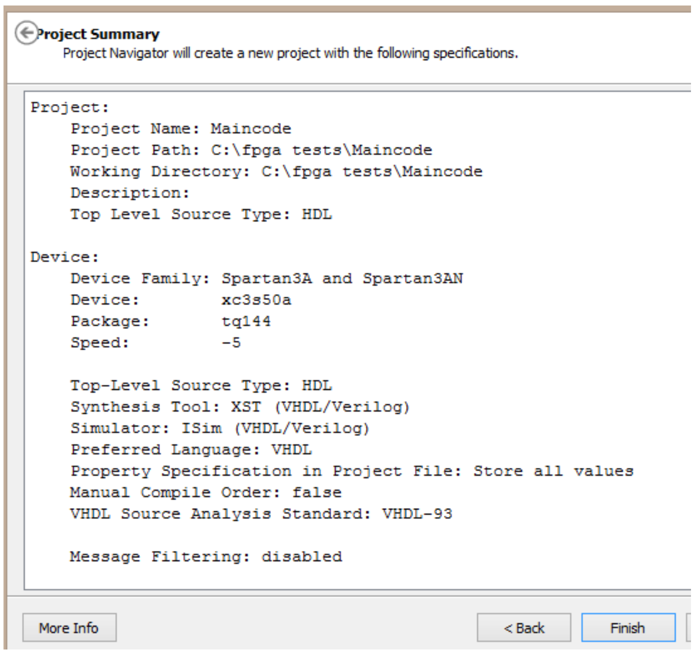
Project Settings

Specify device and project properties.
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan6
Device	XC6SLX16
Package	CSG324
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93

Figure 2.18 Design Parameters

- Project summery window occurs. Click on Finish

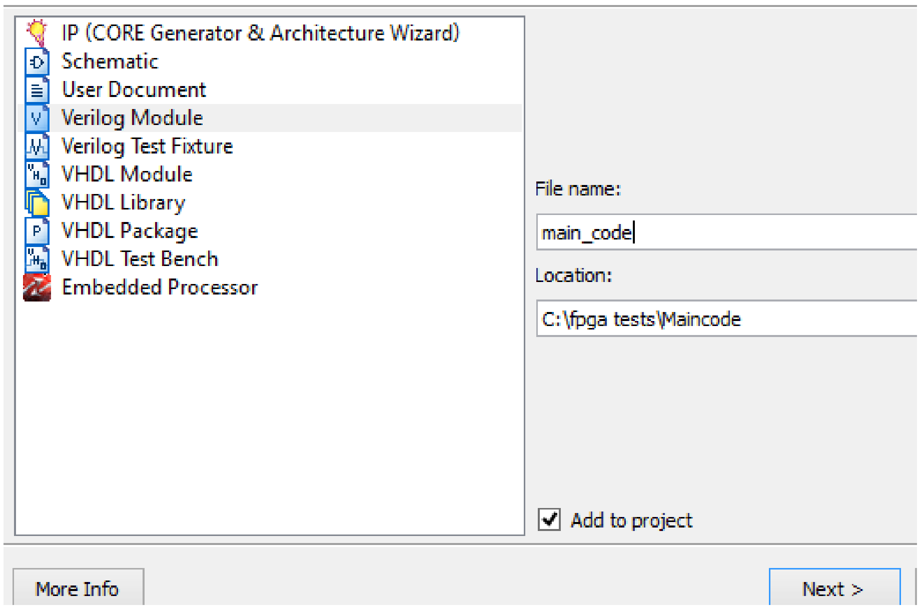


Project Summary
Project Navigator will create a new project with the following specifications.

```
Project:
    Project Name: Maincode
    Project Path: C:\fpga tests\Maincode
    Working Directory: C:\fpga tests\Maincode
    Description:
    Top Level Source Type: HDL

Device:
    Device Family: Spartan3A and Spartan3AN
    Device:        xc3s50a
    Package:       tq144
    Speed:         -5

    Top-Level Source Type: HDL
    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISim (VHDL/Verilog)
    Preferred Language: VHDL
    Property Specification in Project File: Store all values
    Manual Compile Order: false
    VHDL Source Analysis Standard: VHDL-93

    Message Filtering: disabled
```

Figure 2.19 Project summary

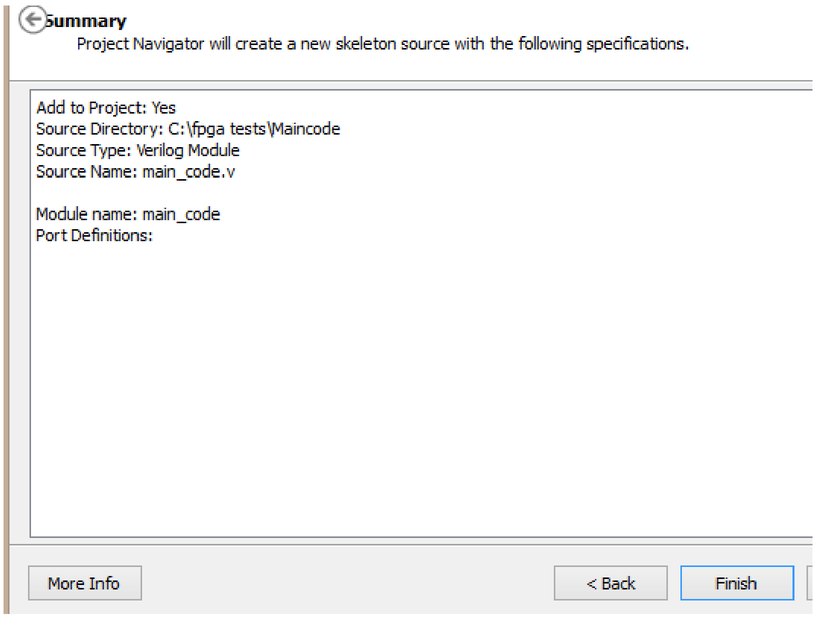
- Click on Project > New Source
- Select Source type is “Verilog Module” and enter the file name (Main_code) and click on Next.



File name: main_code
Location: C:\fpga tests\Maincode
☑ Add to project

Figure 2.20 New source

- Click on Next again. A summary window will pop up and then click on Finish.
- The Project Navigator window will now open with predefined inputs and outputs if set. (In our case we have not set any as we will copy code from repository).



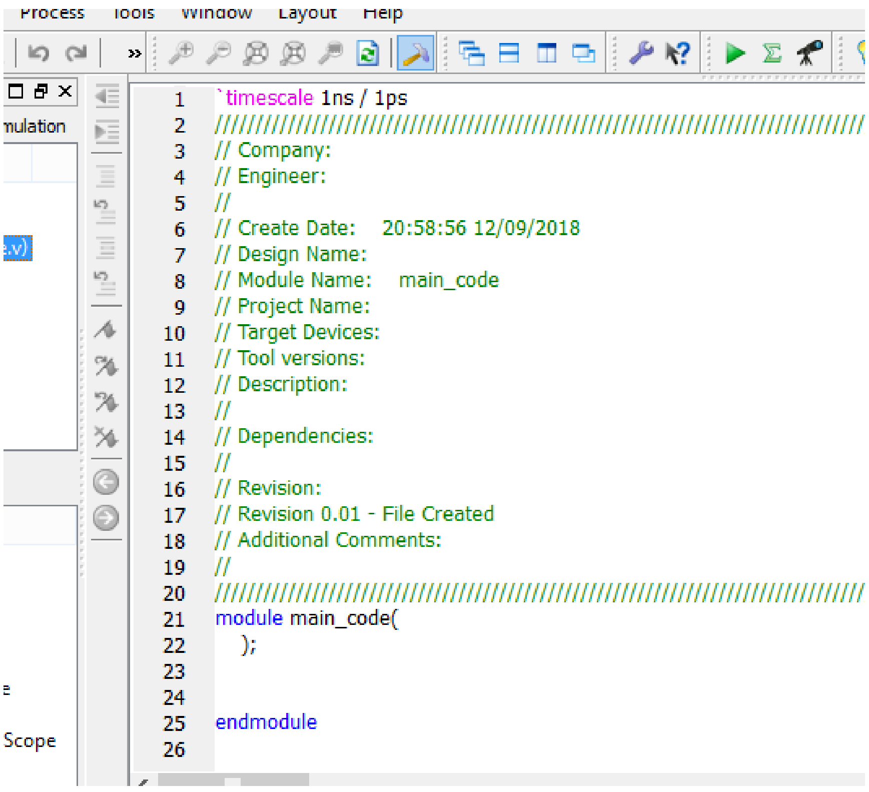
Summary
Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: C:\fpga tests\Maincode
Source Type: Verilog Module
Source Name: main_code.v

Module name: main_code
Port Definitions:

Figure 2.21 Source summary

- Now Go to the repository for the FPGA codes i.e. FPGA_NodeMCU/Codes/FPGA codes/ and download the 3 files and copy them in separate text files.
- Copy paste the "Main code" file into the now open Project Navigator Window after deleting the earlier predefined code lines written in it. Now your top module is ready for FPGA codes.

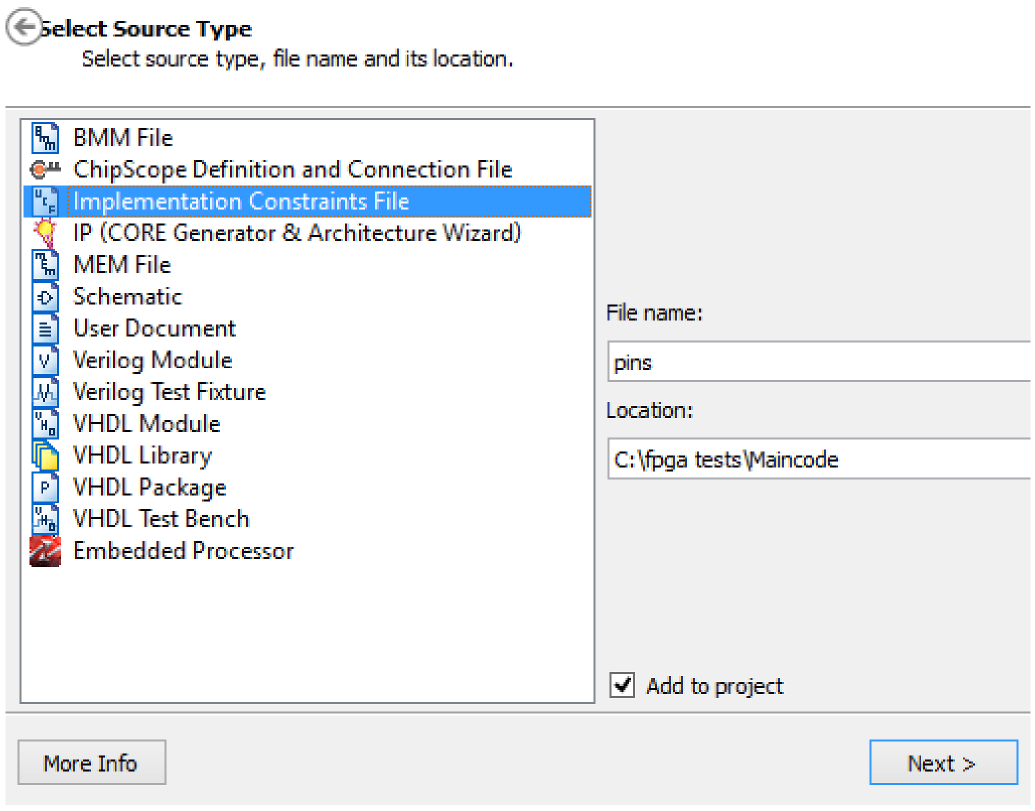


```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    20:58:56 12/09/2018
// Design Name:
// Module Name:    main_code
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module main_code(
    );


endmodule
```

Figure 2.22 Project navigation window

- Follow the above steps to similarly add the "slowClock" module under your top module by again going to New Source and copying the "slowClock" file from the repository.

- Now we have to map the input/output pins on the FPGA board for this project by writing the UCF (User Constraint File) code.
- Now go to Project > New Source. Select Implementation Constraints file type and enter the file name (e.g. pinout).



Select Source Type

Select source type, file name and its location.

- BMM File
- ChipScope Definition and Connection File
- Implementation Constraints File
- IP (CORE Generator & Architecture Wizard)
- MEM File
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

File name: pins

Location: C:\fpga tests\Maincode

☑ Add to project

Figure 2.23 Constraints file

- Copy the UCF file from the repository into this module as it is.
- Once all the files are copied we can now synthesize the code.
- Open main program and double click on Synthesize – XST. After successful completion of Synthesis, double click on Implement Design. Implement design consists of three parts- Translate, Map, Place and Route.

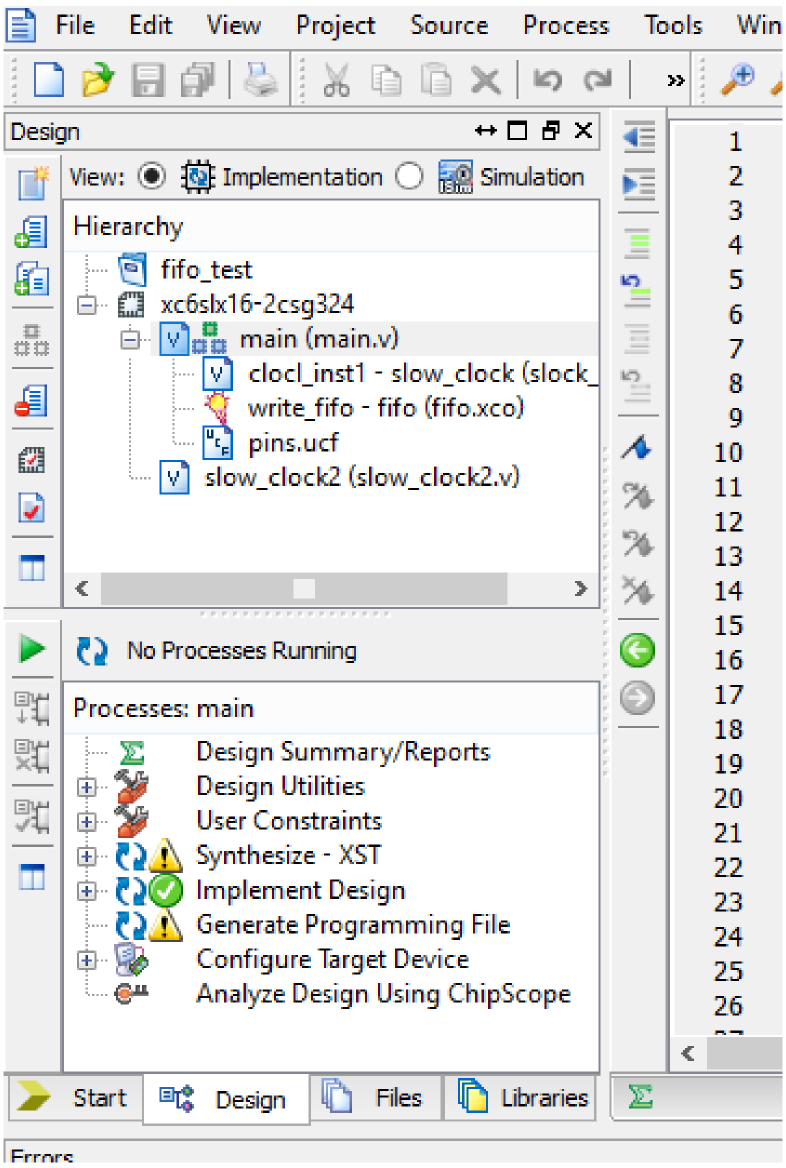


Figure 2.24 Synthesizing the code

- After successful completion of implement design double click on “Generate Programming File”. Programming File Generation produces a bit-stream for Xilinx device configuration.
- After successful generation of bit-stream double click on “Configure Target Device” and a new ISE iMPACT window open.
- Connect the Xilinx board to your PC/Laptop using USB cable.

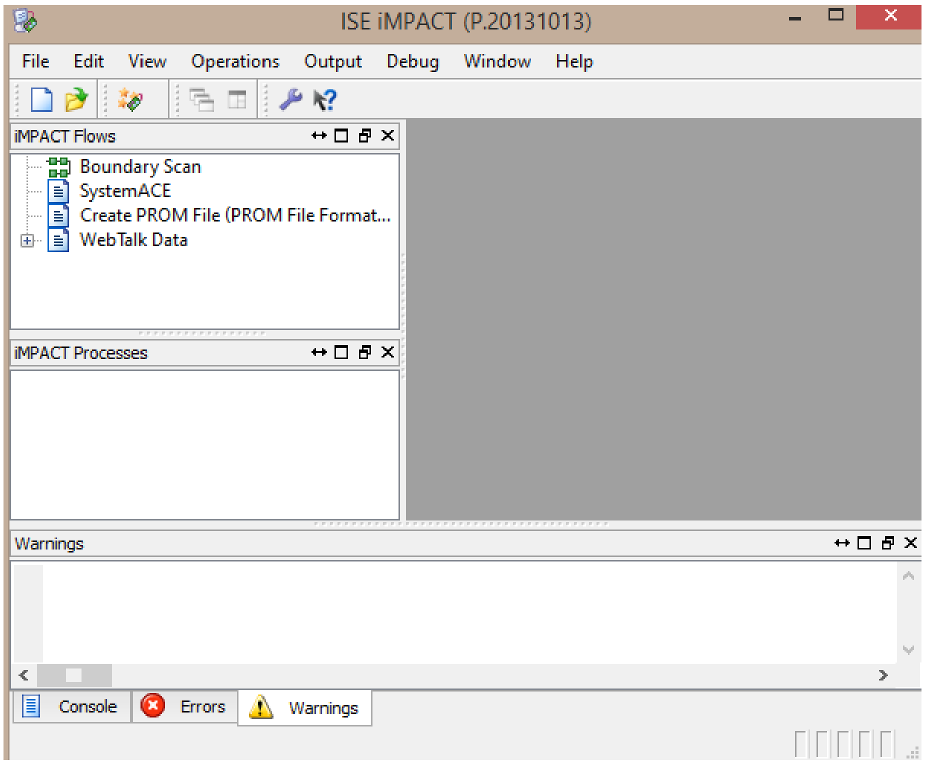
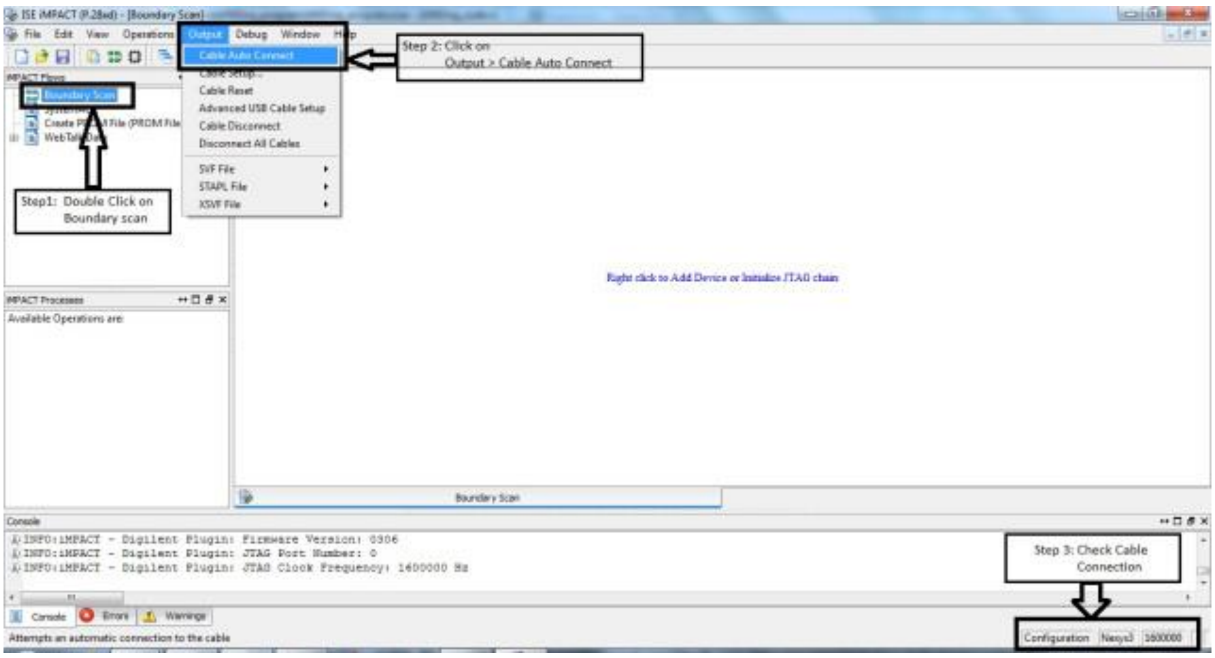


Figure 2.25 ISE Impact window

- Double click on Boundary scan. Check auto cable connection Output > Cable Auto Connect and if cable is connected then Window bottom part looks like step 3 shown in below.



```
INFO:iMPACT - Digilent Plugin: Firmware Version: 0306
INFO:iMPACT - Digilent Plugin: JTAG Port Number: 0
INFO:iMPACT - Digilent Plugin: JTAG Clock Frequency: 1600000 Hz
```

Figure 2.26 Boundary scan

- Click on File > Initialize Chain. After that they ask for “Do you want to continue and assign configuration files(s)?”
- Click on Yes and select the generated bit-stream file.

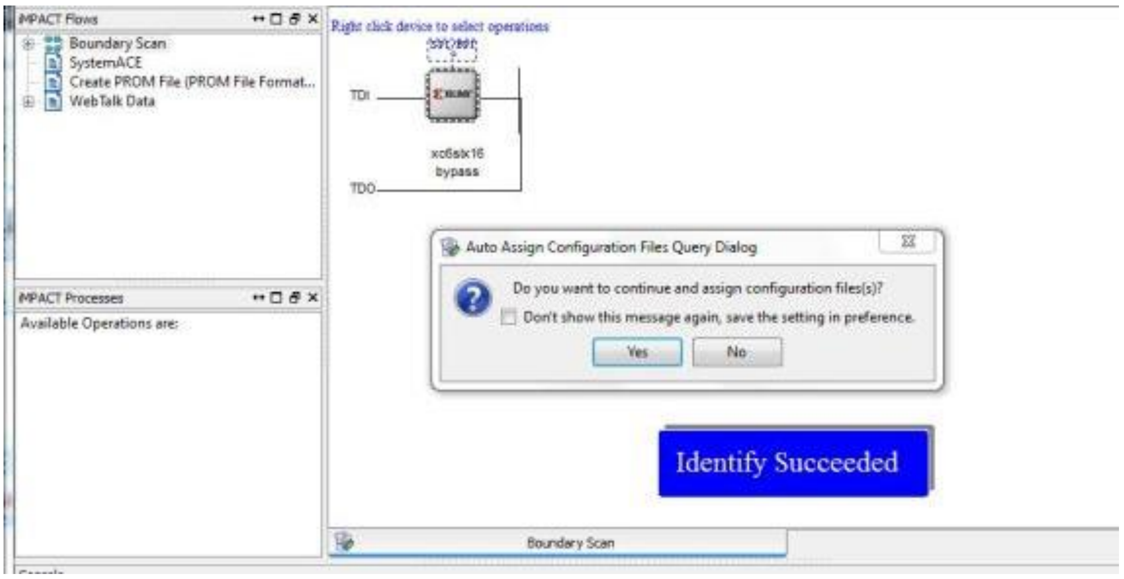
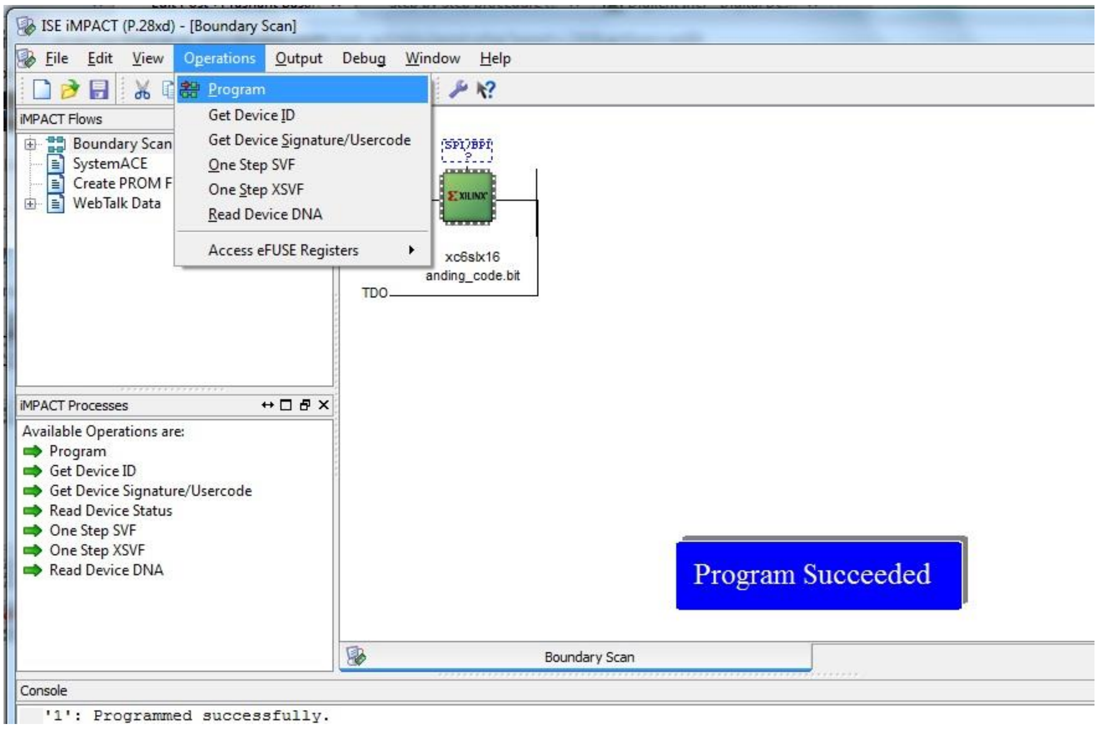

iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File Format...
WebTalk Data
iMPACT Processes
Available Operations are:
Right click device to select operations
SPI/BPI
TDI
TDO
xc6slx16
bypass
Auto Assign Configuration Files Query Dialog
Do you want to continue and assign configuration files(s)?
Don't show this message again, save the setting in preference.
Yes
No
Identify Succeeded
Boundary Scan

Figure 2.27 Identify device

- Click on Open. After that they ask “Do you want to attach an SPI or BPI PROM to this device?” click on No tab. Click on Operations > Program. If Programming successful then they shows Program Succeeded.



```
'1': Programmed successfully.
```

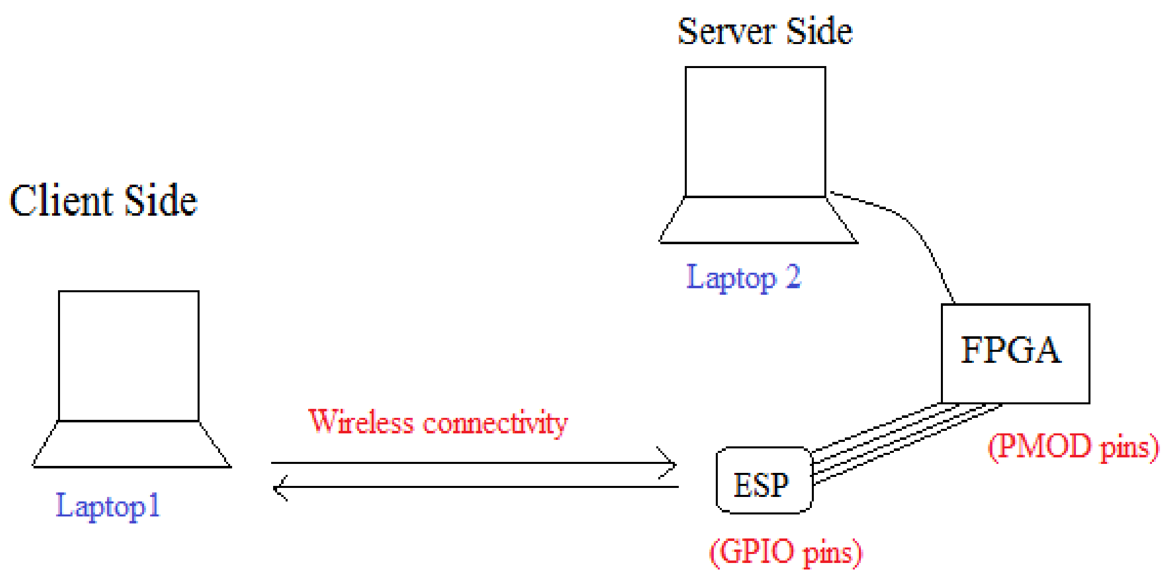
Figure 2.28 Program Board

- Now on the FPGA board two LED’s will light up which indicates the FULL and EMPTY status for the FIFO.
- Press down the first switch (T5) in order to reset the FPGA.
- The FPGA is now ready for instructions from the client and server side.

Chapter 3

3 INTERFACINGS

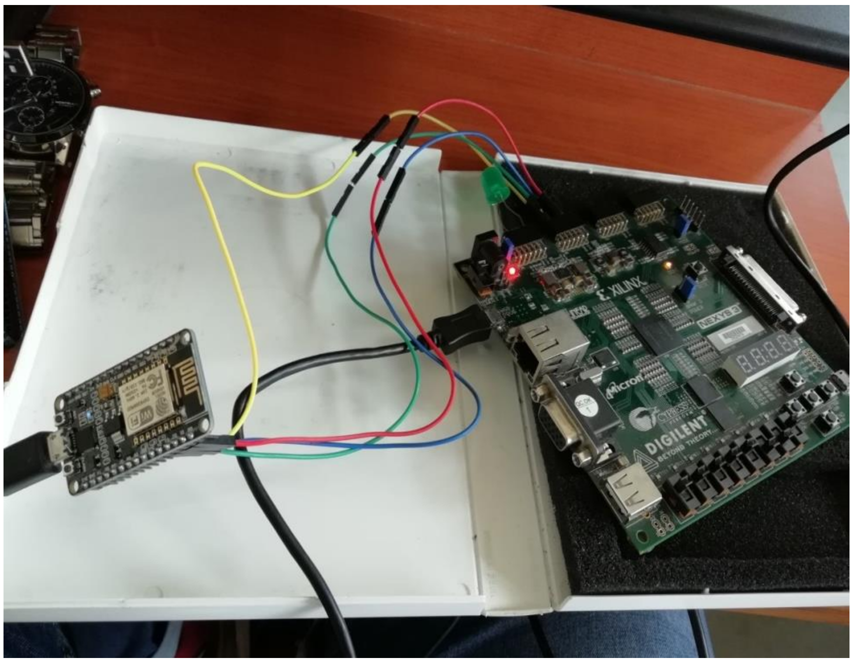
The main interfacings in this project is only at the server side as client side there is only a PC present.



Server side connections:

The NodeMCU and FPGA board are both connected to PC via USB port.

The NodeMCU and FPGA board are connected through their GPIO and PMOD pins respectively.



The details regarding the details regarding these pins can be found in their respective user manual and datasheets given the repository. Go to FPGA_NodeMCU/Datasheets/ to check out the manuals.

The following table gives us the connections between both the modules:

NodeMCU Module	FPGA Board
D4	M10
D6	K2
D7	K1
D0	T12
D1	V12
D2	N10
D5	P11

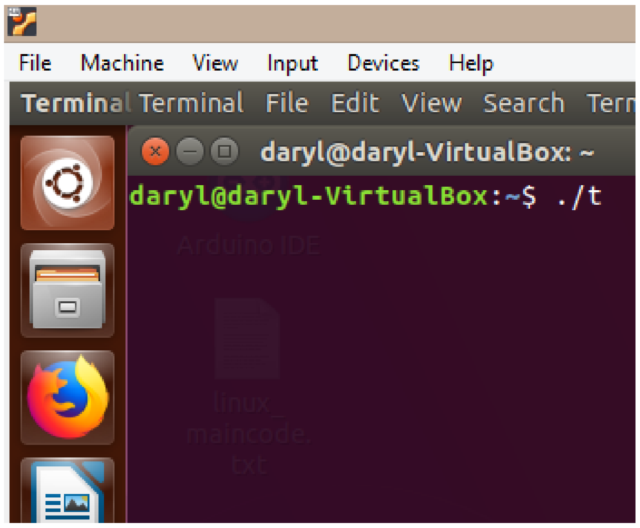
Chapter 4

4 PROJECT IMPLEMENTATION

Once all the interfacings have been done follow the steps at client side to run the project.

Steps to run the project from Client side:

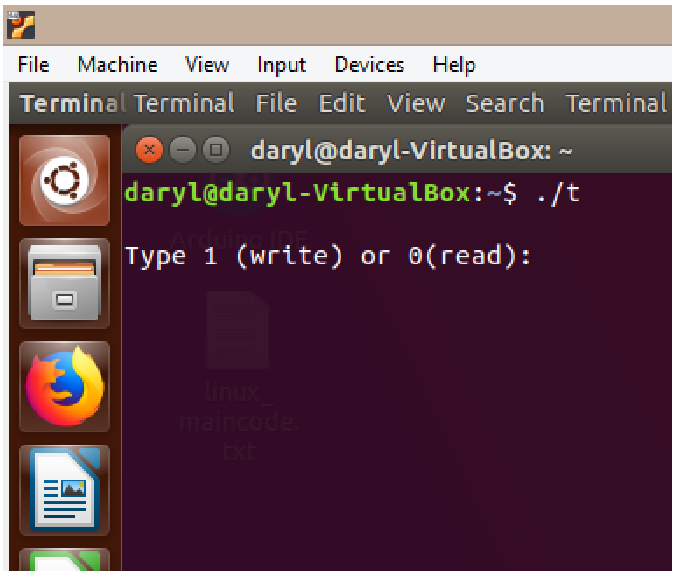
- Open terminal and type ./t


File Machine View Input Devices Help
Terminal Terminal File Edit View Search Tern
daryl@daryl-VirtualBox: ~

```
daryl@daryl-VirtualBox:~$ ./t
```

Figure 4.1 Run the code

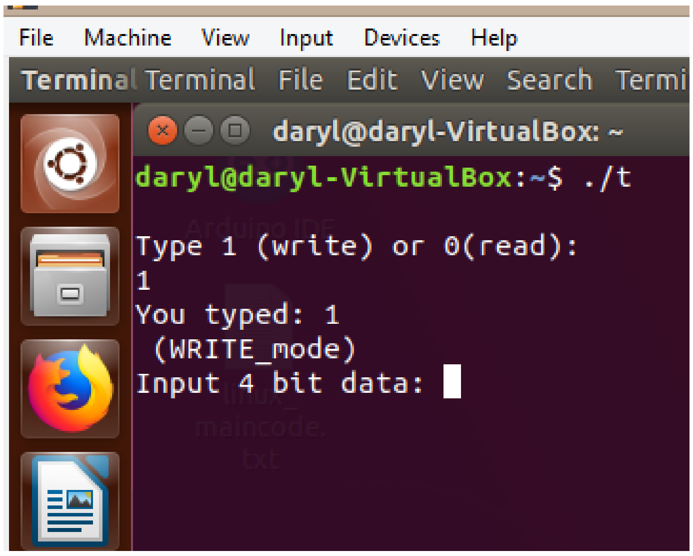
- Now the Socket programming code will run and it will prompt the user to type 1 or 0.


File Machine View Input Devices Help
Terminal Terminal File Edit View Search Terminal
daryl@daryl-VirtualBox: ~

```
daryl@daryl-VirtualBox:~$ ./t

Type 1 (write) or 0(read):
```

Figure 4.2 Question prompted

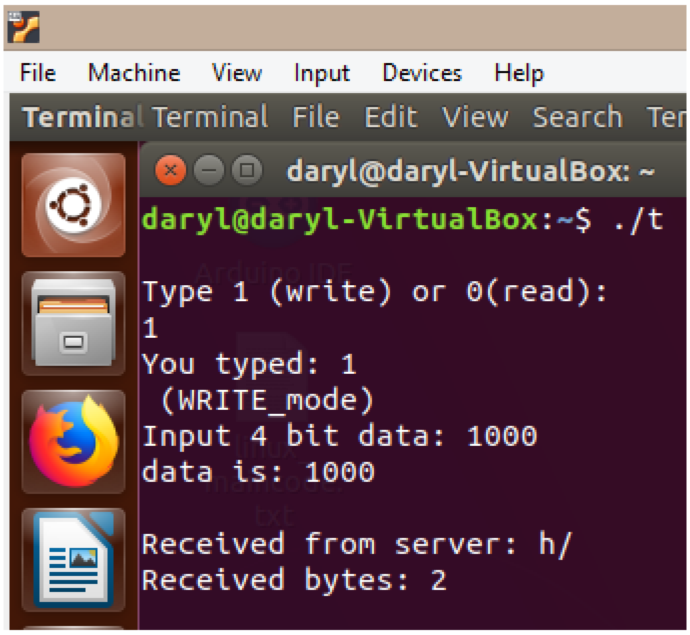
- Type 1 and press enter to write data into the FPGA memory.


File Machine View Input Devices Help
Terminal Terminal File Edit View Search Termi
daryl@daryl-VirtualBox: ~

```
daryl@daryl-VirtualBox:~$ ./t

Type 1 (write) or 0(read):
1
You typed: 1
 (WRITE_mode)
Input 4 bit data: 
```

Figure 4.3 Write command typed (1)

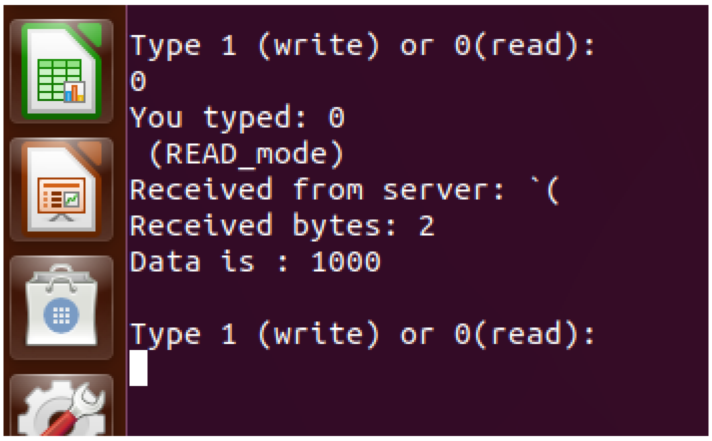
- Now it will ask for the user to input 4 bit binary data.
- Type a 4-bit binary data to be sent to server and written into the FPGA memory.
- As soon as the data is sent you will receive an acknowledgement from server saying data written.


File Machine View Input Devices Help
Terminal Terminal File Edit View Search Ter
daryl@daryl-VirtualBox: ~

```
daryl@daryl-VirtualBox:~$ ./t

Type 1 (write) or 0(read):
1
You typed: 1
 (WRITE_mode)
Input 4 bit data: 1000
data is: 1000

Received from server: h/
Received bytes: 2
```

Figure 4.4 Data sent and ACK received

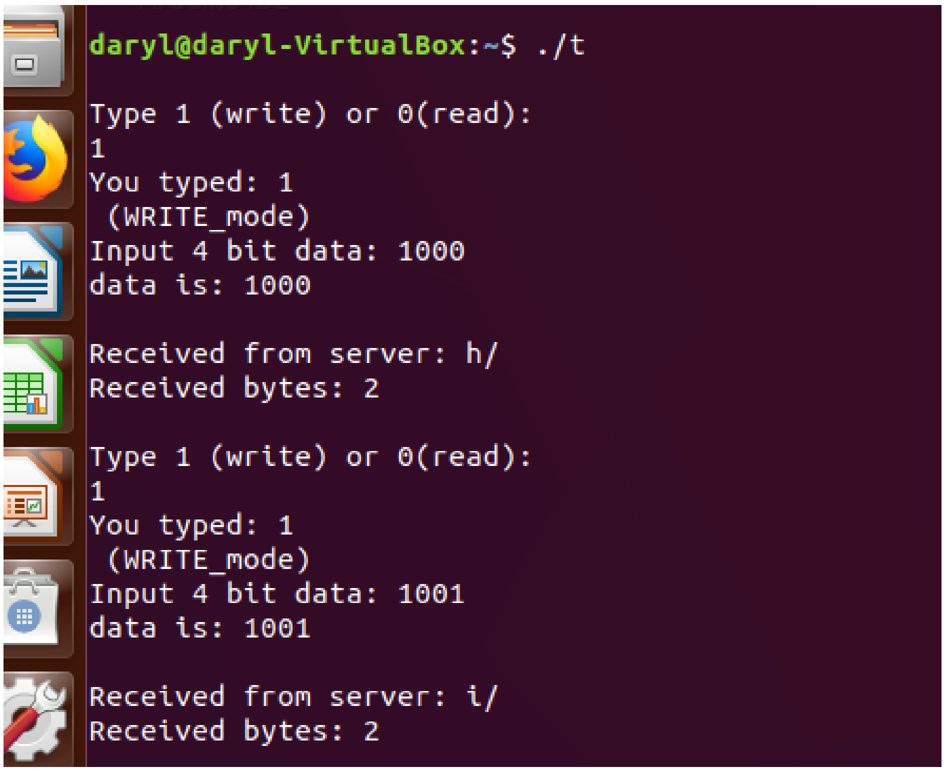
- Now you can again keep sending data or read the data previously sent.
- In order to read the data previously sent type 0 and press enter when prompted.
- Now the first data sent by the client will be available at the client side terminal.



```
Type 1 (write) or 0(read):
0
You typed: 0
 (READ_mode)
Received from server: `(
Received bytes: 2
Data is : 1000

Type 1 (write) or 0(read):
```

Figure 4.5 Data received back

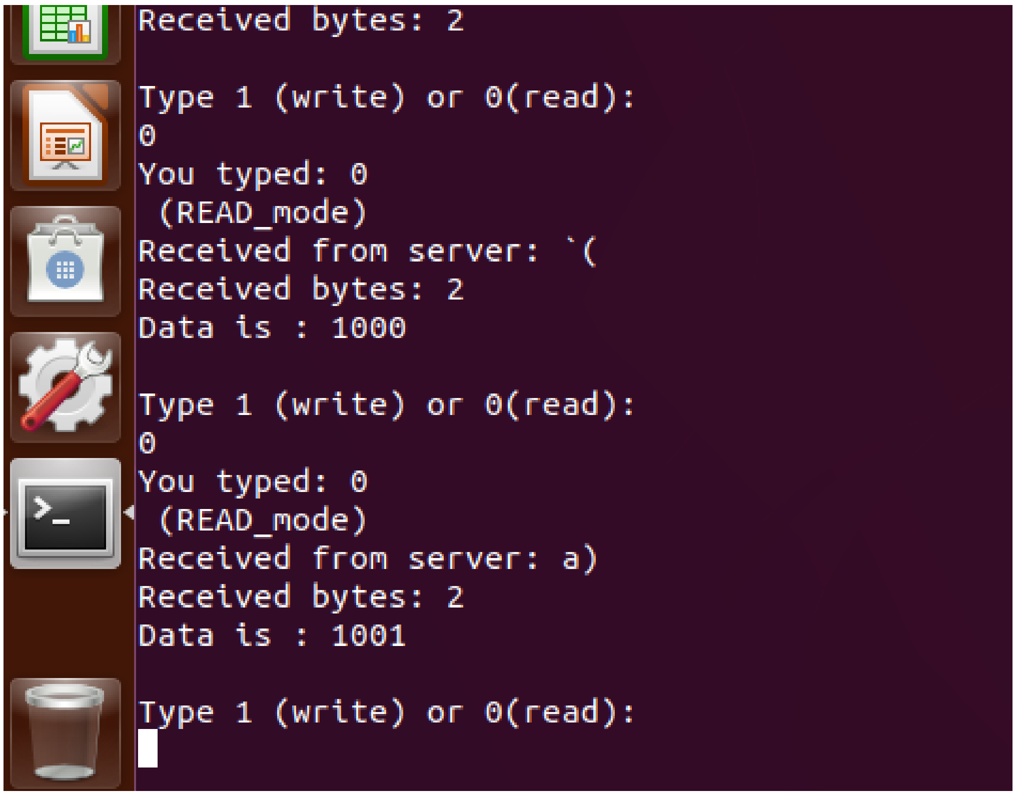
- If you have sent more than one data from client side, repeatedly using the Read operation will show the data in FIFO format (First In First Out).
- Let us now send two data to the server.



```
daryl@daryl-VirtualBox:~$ ./t

Type 1 (write) or 0(read):
1
You typed: 1
 (WRITE_mode)
Input 4 bit data: 1000
data is: 1000

Received from server: h/
Received bytes: 2

Type 1 (write) or 0(read):
1
You typed: 1
 (WRITE_mode)
Input 4 bit data: 1001
data is: 1001

Received from server: i/
Received bytes: 2
```

Figure 4.6 Sending two data

- Now let us try to read them back.
- The server will send the 1000 data back first and then the 1001 data as it is based on FIFO memory.



```
Received bytes: 2

Type 1 (write) or 0(read):
0
You typed: 0
 (READ_mode)
Received from server: `(
Received bytes: 2
Data is : 1000

Type 1 (write) or 0(read):
0
You typed: 0
 (READ_mode)
Received from server: a)
Received bytes: 2
Data is : 1001

Type 1 (write) or 0(read):
```

Figure 4.7 Both data received at the client side