Kinect Based Module Development

E-Yantra Summer Internship Program

Team: Akhil Jain Anshul Panjabi
Mentors: Saurav S. Thyagarajan R.

July 8, 2014

Contents

1 Installation of necessary software **5**
1.1 **Installation of Kinect for Windows SDK (Windows)** . . . 5
1.1.1 **System Requirements** . . . 5
1.1.2 **Installation Procedure** . . . 5
1.2 **Installation of Openkinect's Libfreenect (Ubuntu)** . . . 11
1.2.1 **System Requirements** . . . 11
1.2.2 **Installation Procedure** . . . 11

2 Establishing Unicast connection between 2 XBee modules **20**
2.1 **Installation of X-CTU (Windows)** . . . 20
2.1.1 **System Requirements** . . . 20
2.1.2 **Installation Procedure** . . . 20
2.1.3 **Configuring 2 XBee modules for unicast connection** . . . 23

3 Set up Process on Visual Studio **29**
3.1 **Introduction** . . . 29
3.1.1 **System Requirements** . . . 29
3.1.2 **Set up Procedure** . . . 29

4 Experiments - The fun part **37**
Experiments on the windows platform . . . 38
4.1 **Depth tracking on screen** . . . 38
4.1.1 **Aim of the experiment** . . . 38
4.1.2 **Components required** . . . 38
4.1.3 **Experiment description** . . . 38
4.1.4 **Instructions** . . . 38
4.1.5 **Experiment Outputs** . . . 39
4.1.6 **Conclusion** . . . 39
4.2 **Depth tracking on firebird** . . . 40
4.2.1 **Aim of the experiment** . . . 40
4.2.2 **Components required** . . . 40
4.2.3 **Experiment description** . . . 40
4.2.4 **Instructions** . . . 40
4.2.5 **Experiment Outputs** . . . 41
4.2.6 **Conclusion** . . . 41
4.3 **Camera Fundamentals** . . . 42
4.3.1 **Aim of the experiment** . . . 42
4.3.2 **Components required** . . . 42
4.3.3 **Experiment description** . . . 42
4.3.4 **Instructions** . . . 42
4.3.5 **Experiment Outputs** . . . 42
4.3.6 **Conclusion** . . . 43
4.4 **Skeleton Tracking Fundamentals** . . . 44
4.4.1 **Aim of the experiment** . . . 44
4.4.2 **Components required** . . . 44
4.4.3 **Experiment description** . . . 44
4.4.4 **Instructions** . . . 44
4.4.5 **Experiment Outputs** . . . 44
4.4.6 **Conclusion** . . . 45

4.5 **Skeleton Tracking Firebird Left Right** . . . 46
4.5.1 **Aim of the experiment** . . . 46
4.5.2 **Components required** . . . 46
4.5.3 **Experiment description** . . . 46
4.5.4 **Instructions** . . . 46
4.5.5 **Experiment Outputs** . . . 47
4.5.6 **Conclusion** . . . 47
4.6 **Skeleton Tracking Firebird Front Back** . . . 48
4.6.1 **Aim of the experiment** . . . 48
4.6.2 **Components required** . . . 48
4.6.3 **Experiment description** . . . 48
4.6.4 **Instructions** . . . 48
4.6.5 **Experiment Outputs** . . . 49
4.6.6 **Conclusion** . . . 49
4.7 **Skeleton Tracking angle between joints** . . . 50
4.7.1 **Aim of the experiment** . . . 50
4.7.2 **Components required** . . . 50
4.7.3 **Experiment description** . . . 50
4.7.4 **Instructions** . . . 50
4.7.5 **Experiment Outputs** . . . 51
4.7.6 **Conclusion** . . . 51
4.8 **Skeleton Tracking Hoverbutton Firebird** . . . 52
4.8.1 **Aim of the experiment** . . . 52
4.8.2 **Components required** . . . 52
4.8.3 **Experiment description** . . . 52
4.8.4 **Instructions** . . . 52
4.8.5 **Experiment Outputs** . . . 53
4.8.6 **Conclusion** . . . 53
4.9 **Skeleton Tracking Complete** . . . 54
4.9.1 **Aim of the experiment** . . . 54
4.9.2 **Components required** . . . 54
4.9.3 **Experiment description** . . . 54
4.9.4 **Instructions** . . . 54
4.9.5 **Experiment Outputs** . . . 55
4.9.6 **Conclusion** . . . 55
4.10 **Speech Recognition** . . . 56
4.10.1 **Aim of the experiment** . . . 56
4.10.2 **Components required** . . . 56
4.10.3 **Experiment description** . . . 56
4.10.4 **Instructions** . . . 56
4.10.5 **Experiment Outputs** . . . 57
4.10.6 **Conclusion** . . . 57
4.11 **Tilt Demo** . . . 58
4.11.1 **Aim of the experiment** . . . 58
4.11.2 **Components required** . . . 58
4.11.3 **Experiment description** . . . 58
4.11.4 **Instructions** . . . 58
4.11.5 **Experiment Outputs** . . . 58
4.11.6 **Conclusion** . . . 59
Experiments on the linux platform . . . 61
4.12 **Depth tracking on screen** . . . 61
4.12.1 **Aim of the experiment** . . . 61
4.12.2 **Components required** . . . 61
4.12.3 **Experiment description** . . . 61
4.12.4 **Instructions** . . . 61
4.12.5 **Experiment Outputs** . . . 62
4.12.6 **Conclusion** . . . 62
4.13 **Depth tracking with firebird** . . . 63
4.13.1 **Aim of the experiment** . . . 63
4.13.2 **Components required** . . . 63
4.13.3 **Experiment description** . . . 63

4.13.4 **Instructions** . . . 63
4.13.5 **Experiment Outputs** . . . 64
4.13.6 **Conclusion** . . . 64
4.14 **Tilt Demo** . . . 65
4.14.1 **Aim of the experiment** . . . 65
4.14.2 **Components required** . . . 65
4.14.3 **Experiment description** . . . 65
4.14.4 **Instructions** . . . 65
4.14.5 **Experiment Outputs** . . . 65
4.14.6 **Conclusion** . . . 66

Chapter 1

Installation of necessary software

1.1 Installation of Kinect for Windows SDK (Windows)

Hello world ! This is a quick installation tutorial for the Microsoft's Kinect for Windows SDK. The Kinect for Windows Software Development Kit (SDK) enables developers to create applications that support gestures and voice recognition, using kinect sensor technology. Refer to [1].

Note : At the time of making of this manual, the latest version was Kinect for windows v.1.8.0 The same was used for experimenting with kinect.

1.1.1 System Requirements

- Hardware Requirements
  - 32-bit (x86) or 64-bit (x64) processor
  - Dual-core 2.66-GHz or faster processor
  - Dedicated USB 2.0 bus
  - 2 GB RAM
  - A Microsoft Kinect for Windows (or Xbox) sensor

- Software Requirements
  - Visual Studio 2010, or Visual Studio 2012. The free Express editions can be downloaded from Microsoft Visual Studio 2010 Express or Microsoft Visual Studio 2012 Express.
  - .NET Framework 4 (installed with Visual Studio 2010), or .NET Framwork 4.5 (installed with Visual Studio 2012)
  - To develop speech-enabled Kinect for Windows Applications, you must install the Microsoft Speech Platform SDK v11

1.1.2 Installation Procedure

Step 1 :

1a) First the Microsoft's Kinect for windows SDK has to be downloaded. Download the same. 1b) Double click on the **KinectSDK-v1.8-Setup.exe** from the download location to begin the installation. Visit http://www.microsoft.com/en-in/download/details.aspx?id=40278.
Shown in Figures 1.1 and 1.2 respectively.

Step 2 :

Accept the terms and conditions and click install. Shown in Figure 1.3.

Step 3 :

The installation procedure will complete on its own. Shown in Figure 1.4.

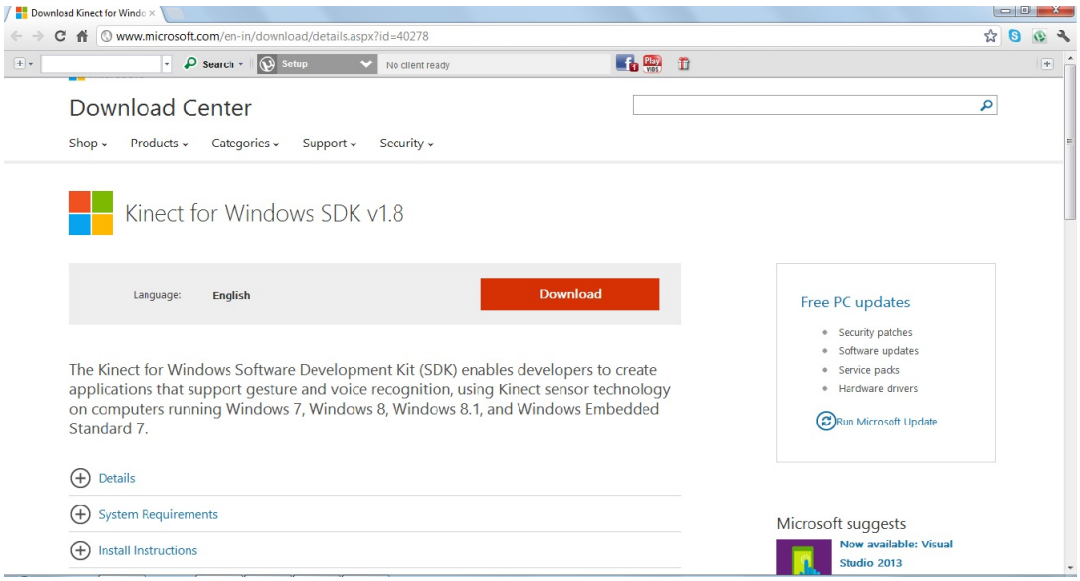


Figure 1.1: Step 1a - Microsoft's Kinect for Windows SDK website

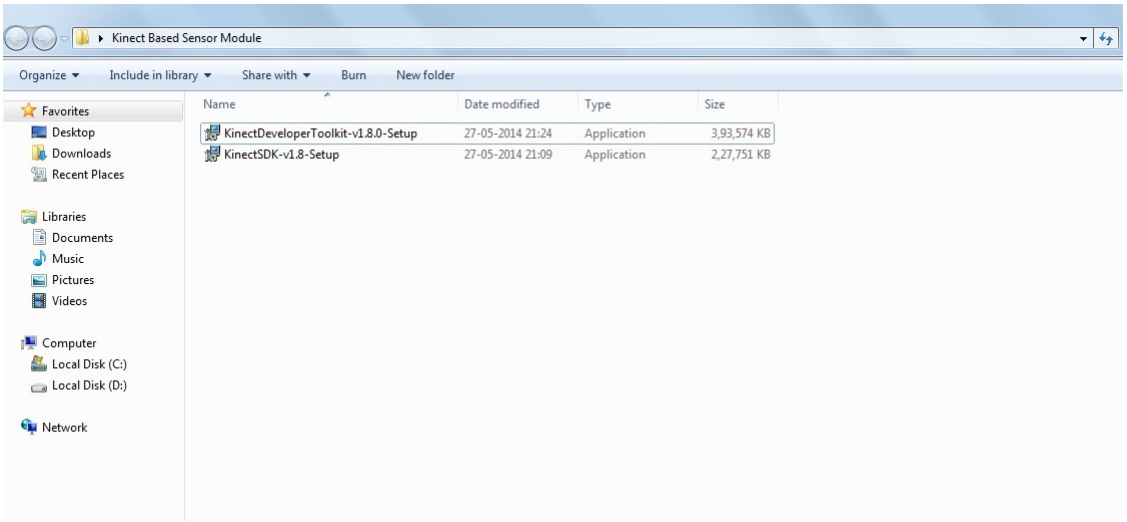


Figure 1.2: Step 1b - The downloaded setup files

Step 4 :

After the installation of the SDK, download the Microsoft's Kinect for windows toolkit from the link in the window. Shown in Figure 1.5.

Step 5 :

Double click on the **KinectDeveloperToolkit-v1.8.0-Setup.exe** from the download location to begin the installation. Accept the terms and conditions and click install. Shown in Figure 1.6.

Step 6 :

The installation procedure will complete on its own. Shown in Figure 1.7.

Step 7 :

Wait for the installation to complete. Shown in Figure 1.8.

Step 8 :

Verify the installation by opening the Developer Toolkit Browser v.1.8.0 Shown in Figure 1.9.

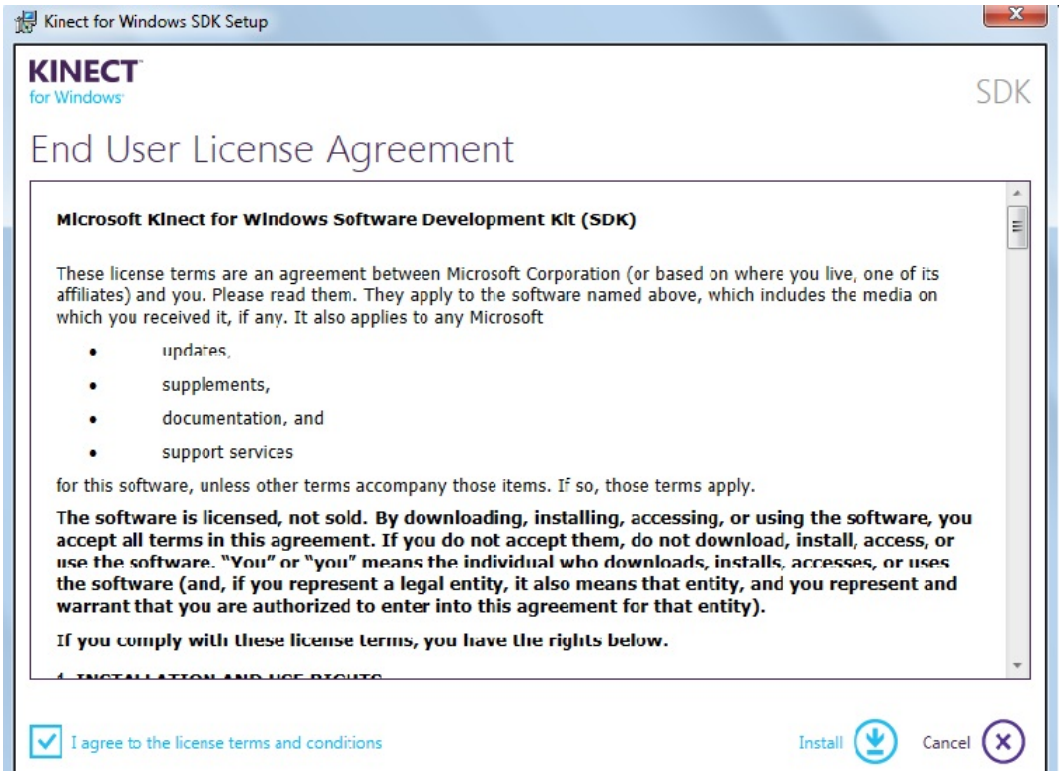


Figure 1.3: Step 2 - Kinect for windows SDK setup

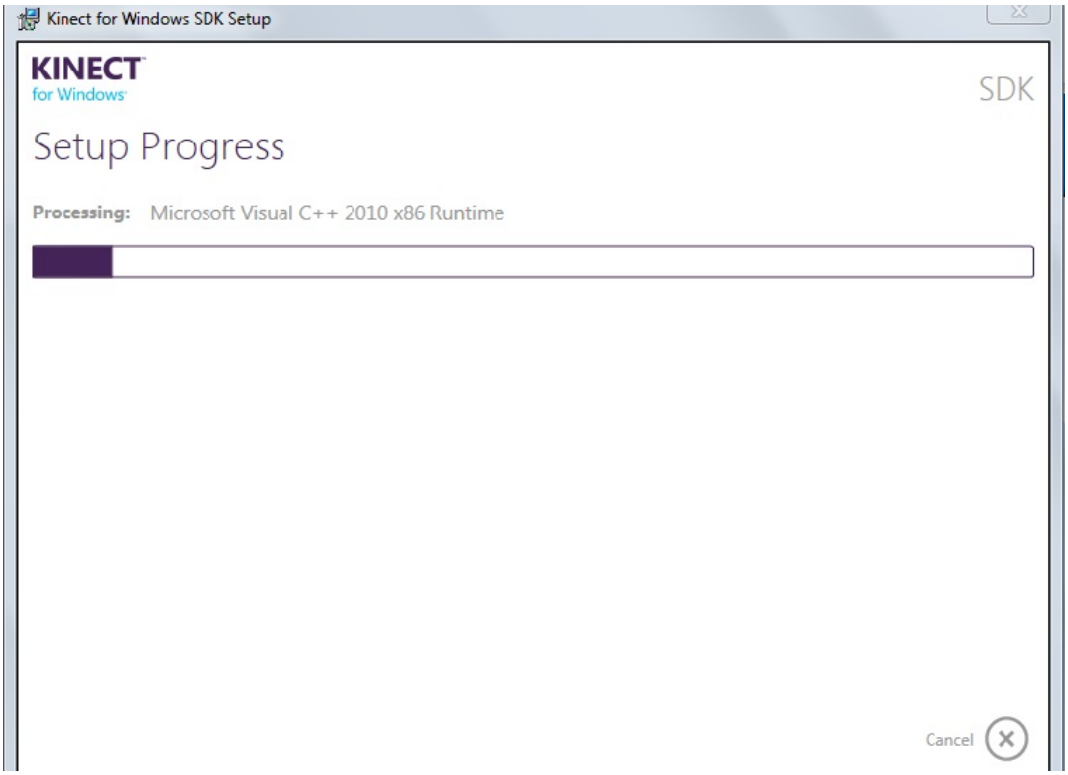


Figure 1.4: Step 3 - Setup process will complete on its own

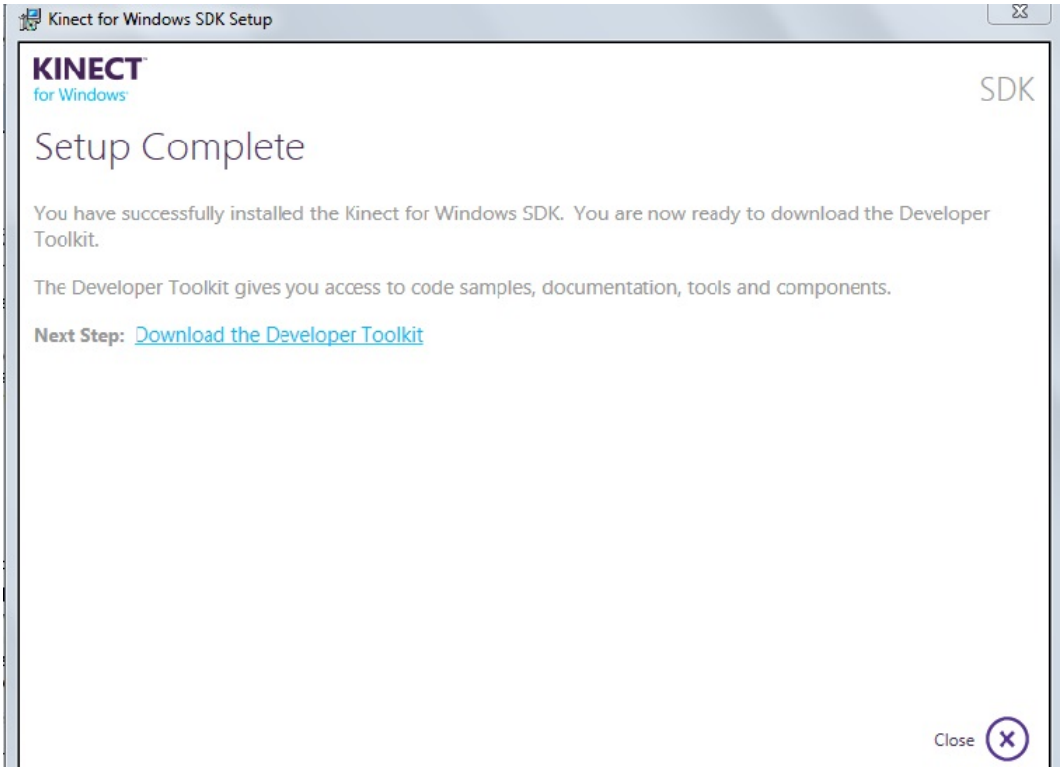


Figure 1.5: Step 4 - Click on "Download the developer toolkit"

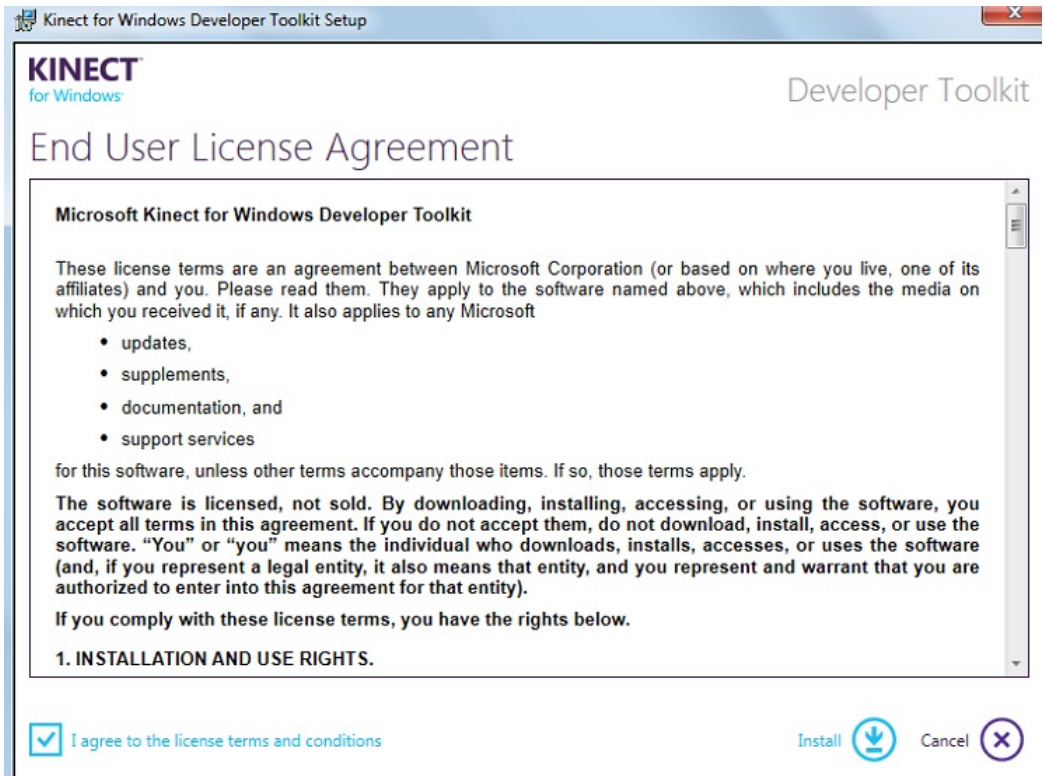


Figure 1.6: Step 5 - Kinect for Windows Developer toolkit setup

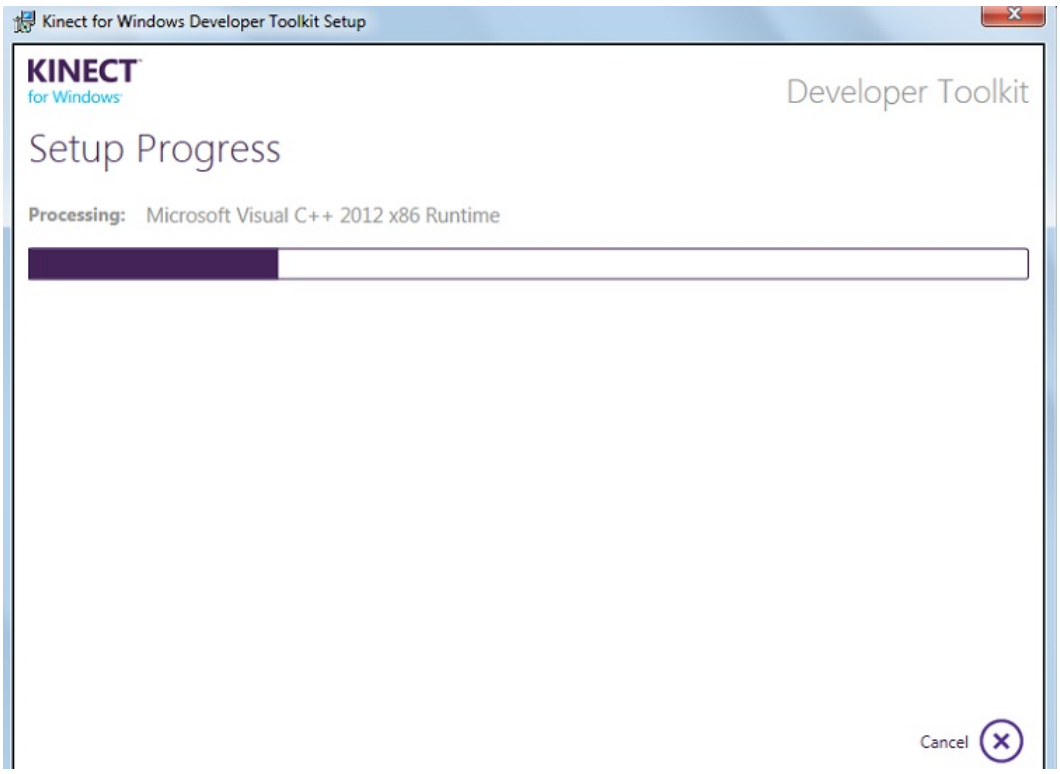


Figure 1.7: Step 6 - The toolkit setup will complete on its own

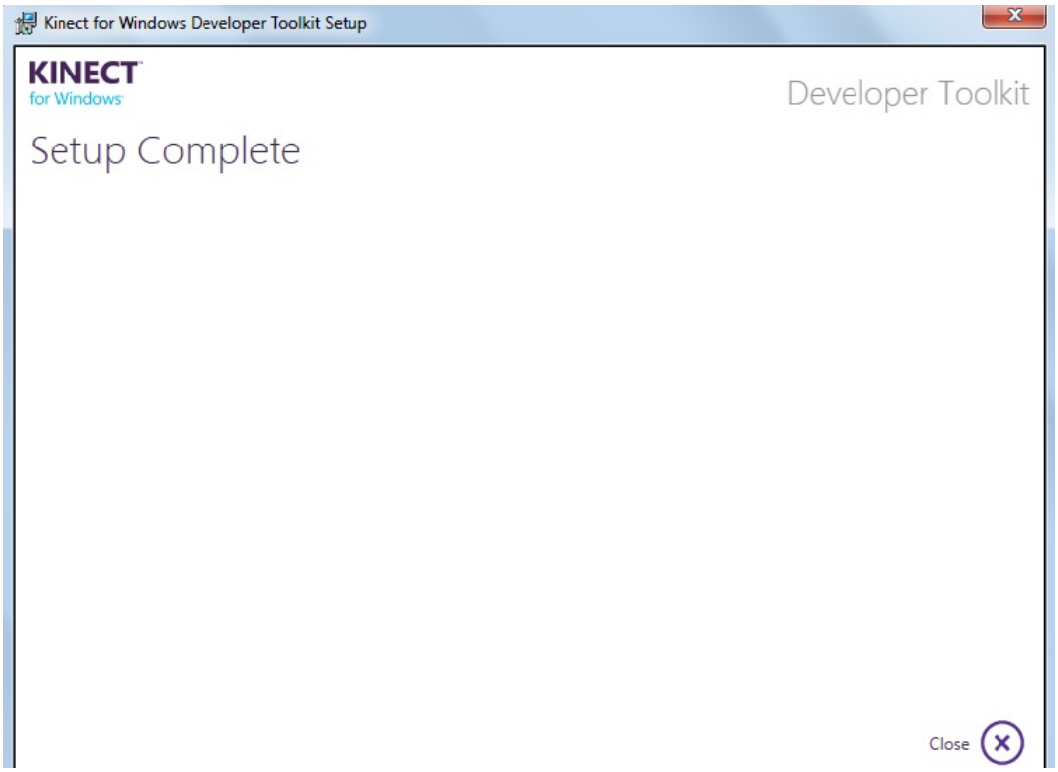


Figure 1.8: Step 7 - The setup is complete

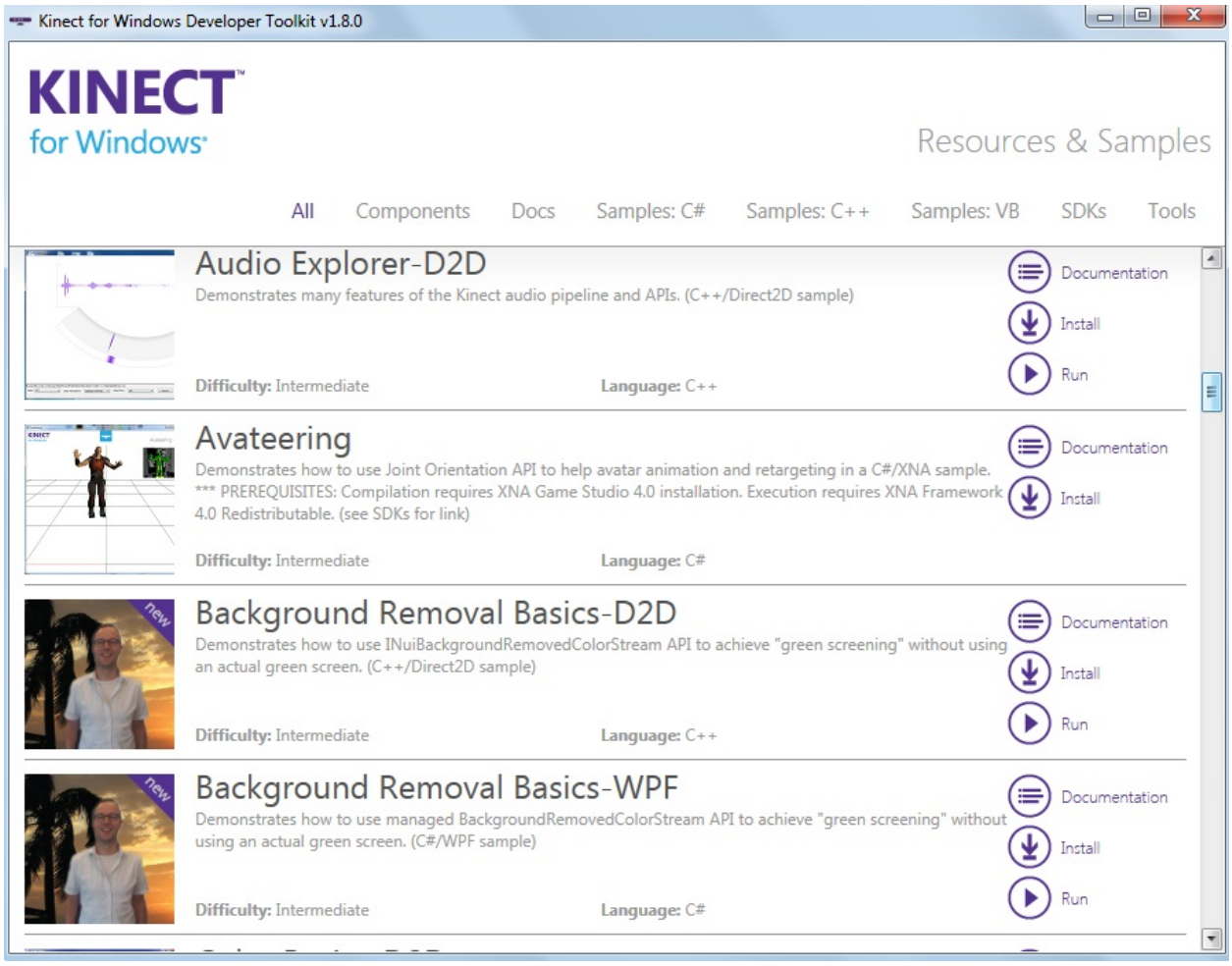


Figure 1.9: Step 8 - Kinect for windows Developer Toolkit v1.8.0

1.2 Installation of Openkinect's Libfreenect (Ubuntu)

Hello world ! This is a quick installation tutorial for the Openkinect's Libfreenect on the Linux platform. Ubuntu 14.04 is the platform used in this tutorial. Refer to [2].

1.2.1 System Requirements

- Hardware Requirements
  - 32-bit (x86) or 64-bit (x64) processor
  - Dual-core 2.66-GHz or faster processor
  - Dedicated USB 2.0 bus
  - 2 GB RAM
  - A Microsoft Kinect for Windows (or Xbox) sensor

1.2.2 Installation Procedure

Step 1 :

Open a new terminal and type -

```
sudo apt-get install git-core cmake libglut3-dev pkg-config build-essential libxmu-dev libxi-dev libusb-1.0-0-dev
```

Enter the password when prompted. The installation will complete on its own.

Note: If you getting an error saying apt-get cannot find libglut3, you might be on a newer version of Ubuntu that has freeglut3-* instead of libglut3-*, so your initial apt-get install would look like:

```
sudo apt-get install git-core cmake freeglut3-dev pkg-config build-essential libxmu-dev libxi-dev libusb-1.0-0-devgit
```

Shown in Figure 1.10.

Step 2 :

Clone the Libfreenect file from Github base by typing

```
clone git://github.com/OpenKinect/libfreenect.git
```

At the completion of cloning you will find a folder named **libfreenect** in your home directory.

Shown in Figure 1.11.

Step 3 :

Enter the libfreenect folder by typing -

```
cd libfreenect
```

Shown in Figure 1.12.

Step 4 :

Make a new directory in the libfreenect folder and name it 'build'

```
mkdir build
```

Shown in Figure 1.13.

Step 5 :

Enter the build directory by typing -

```
cd build
```

Shown in Figure 1.14.

Step 6 :

6a) Now "make" all the projects by typing -

```
cmake ..
```

6b) and then type

```
make
```

Shown in Figures 1.15 and 1.16 respectively

Step 7 :

Next Type -

```
sudo make install
```

Shown in Figure 1.17.

Step 8 :

Configure all the libraries by typing -

```
sudo ldconfig /usr/local/lib64/
```

Shown in Figure 1.18.

Step 9 :

To use Kinect as a non-root user type the following -

```
sudo adduser $USER video
```

Shown in Figure 1.19.

Step 10 :

Now for testing the installed software we will run one of the sample projects by typing -

```
sudo freenect-glview
```

Shown in Figure 1.20.

The Ouput is shown in Figure 1.21.

Note :

- The installation procedure for other platforms is available on

  http://openkinect.org/wiki/Getting_Started

- One can also make a file with rules for the Linux device manager:

  ```
  sudo nano /etc/udev/rules.d/51-kinect.rules
  ```

  Copy and paste:

  ```
  # ATTRproduct==”Xbox NUI Motor” SUBSYSTEM==”usb”, ATTRidVendor==”045e”, ATTRid-
  Product==”02b0”, MODE=”0666”
  # ATTRproduct==”Xbox NUI Audio” SUBSYSTEM==”usb”, ATTRidVendor==”045e”, ATTRid-
  Product==”02ad”, MODE=”0666”
  # ATTRproduct==”Xbox NUI Camera” SUBSYSTEM==”usb”, ATTRidVendor==”045e”, ATTRid-
  Product==”02ae”, MODE=”0666”
  # ATTRproduct==”Xbox NUI Motor” SUBSYSTEM==”usb”, ATTRidVendor==”045e”, ATTRid-
  Product==”02c2”, MODE=”0666”
  # ATTRproduct==”Xbox NUI Motor” SUBSYSTEM==”usb”, ATTRidVendor==”045e”, ATTRid-
  Product==”02be”, MODE=”0666”
  # ATTRproduct==”Xbox NUI Motor” SUBSYSTEM==”usb”, ATTRidVendor==”045e”, ATTRid-
  Product==”02bf”, MODE=”0666”
  ```

  Be sure to log out and back in.

Congratulations ! You have just installed the necessary software for your platform to work on the Kinect sensor.

Figure 1.10: Step 1 - Installation

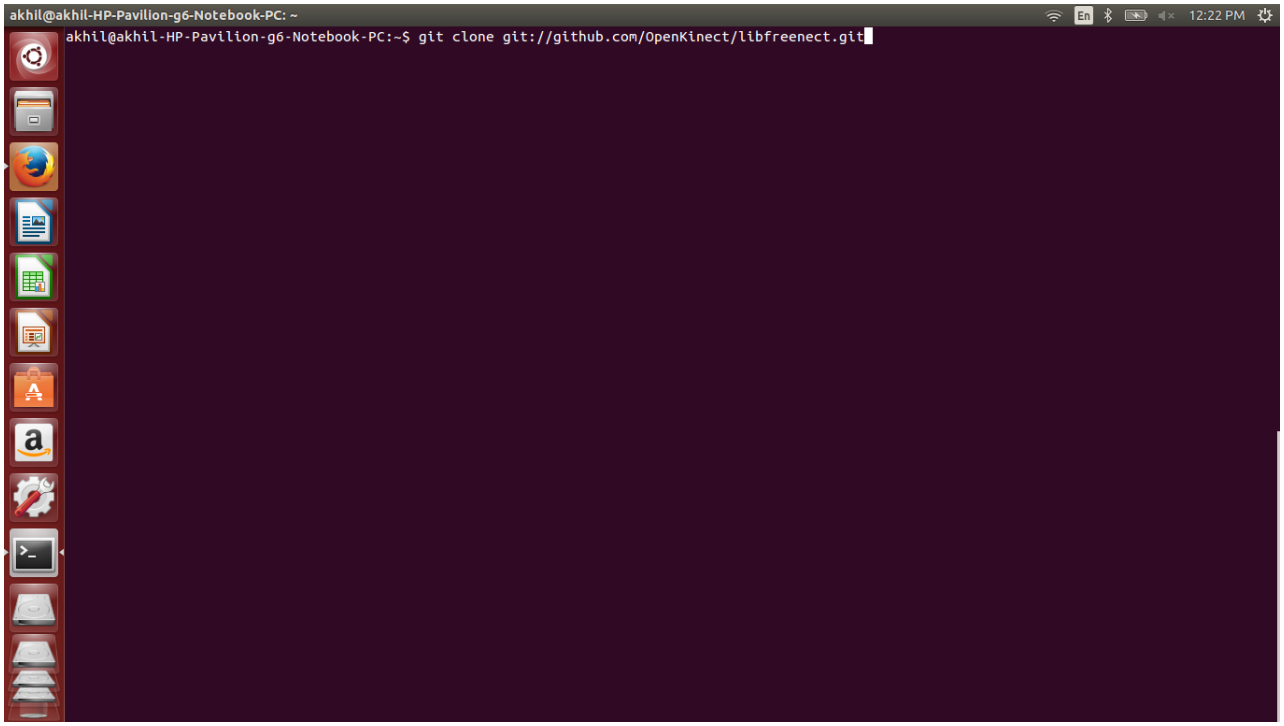


Figure 1.11: Step 2 - Cloning libfreenect

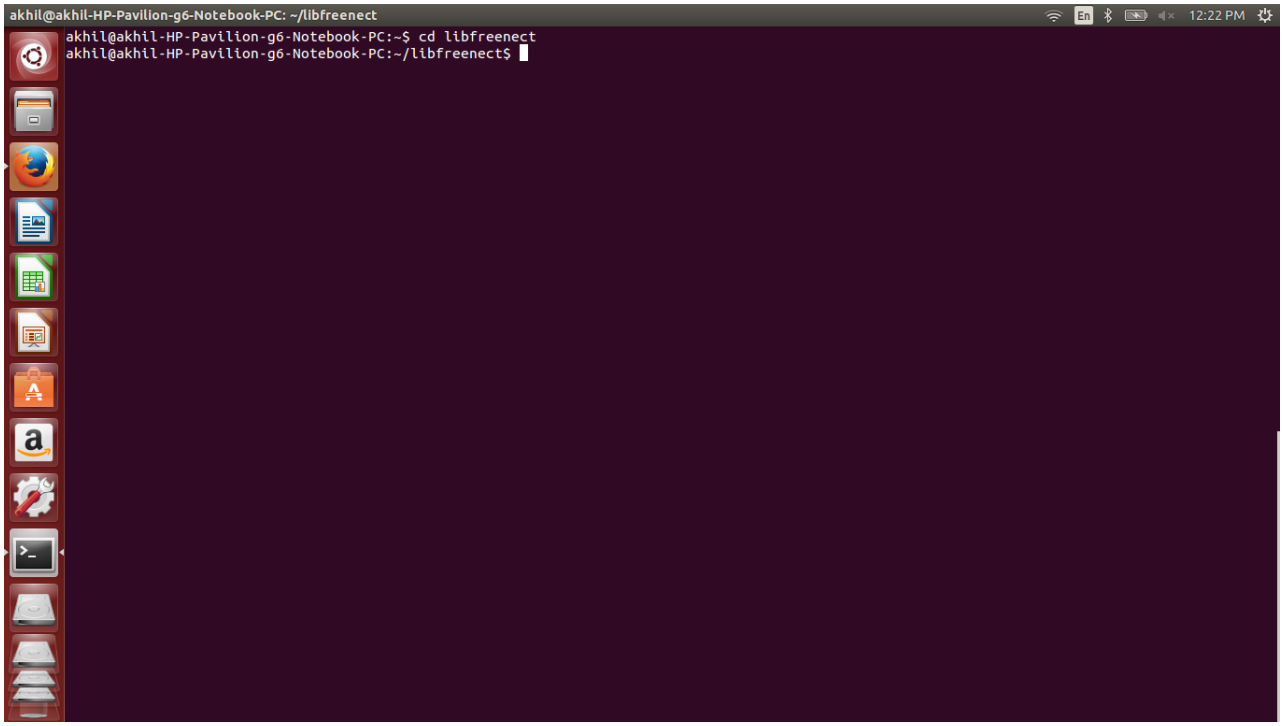


Figure 1.12: Step 3 - Enter the libfreenect folder

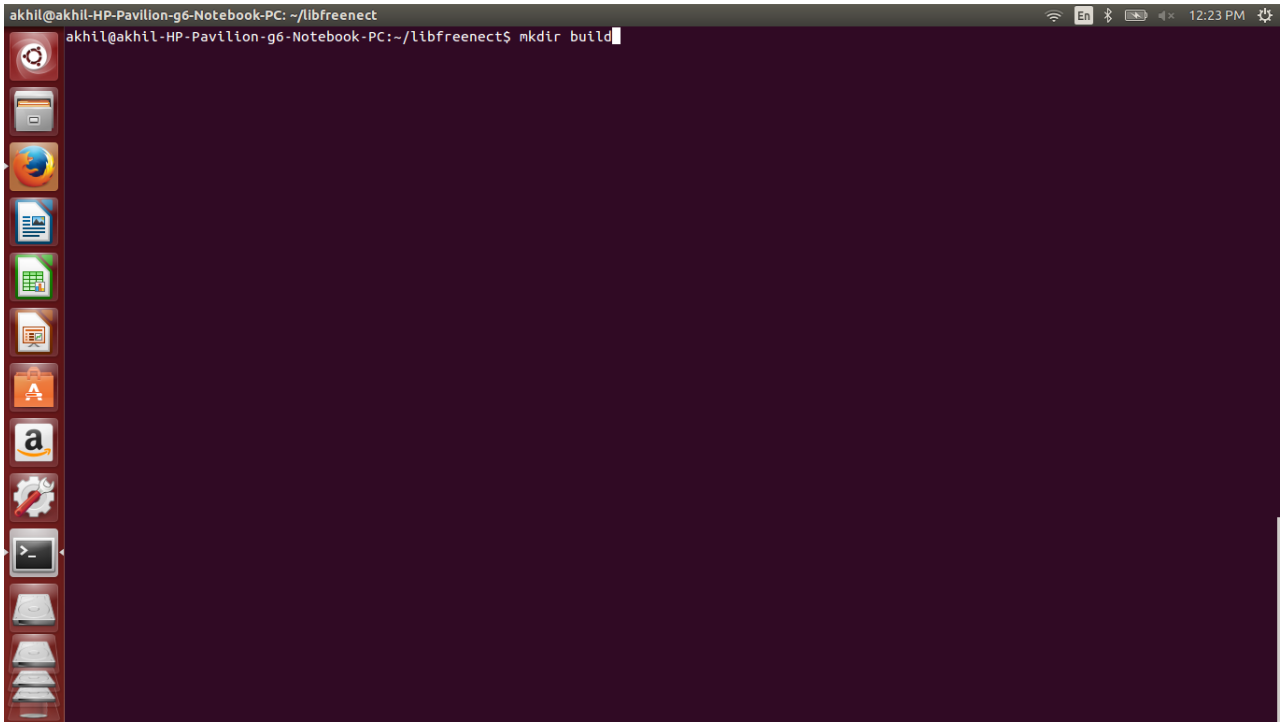


Figure 1.13: Step 4 - Create a new directory named "build"

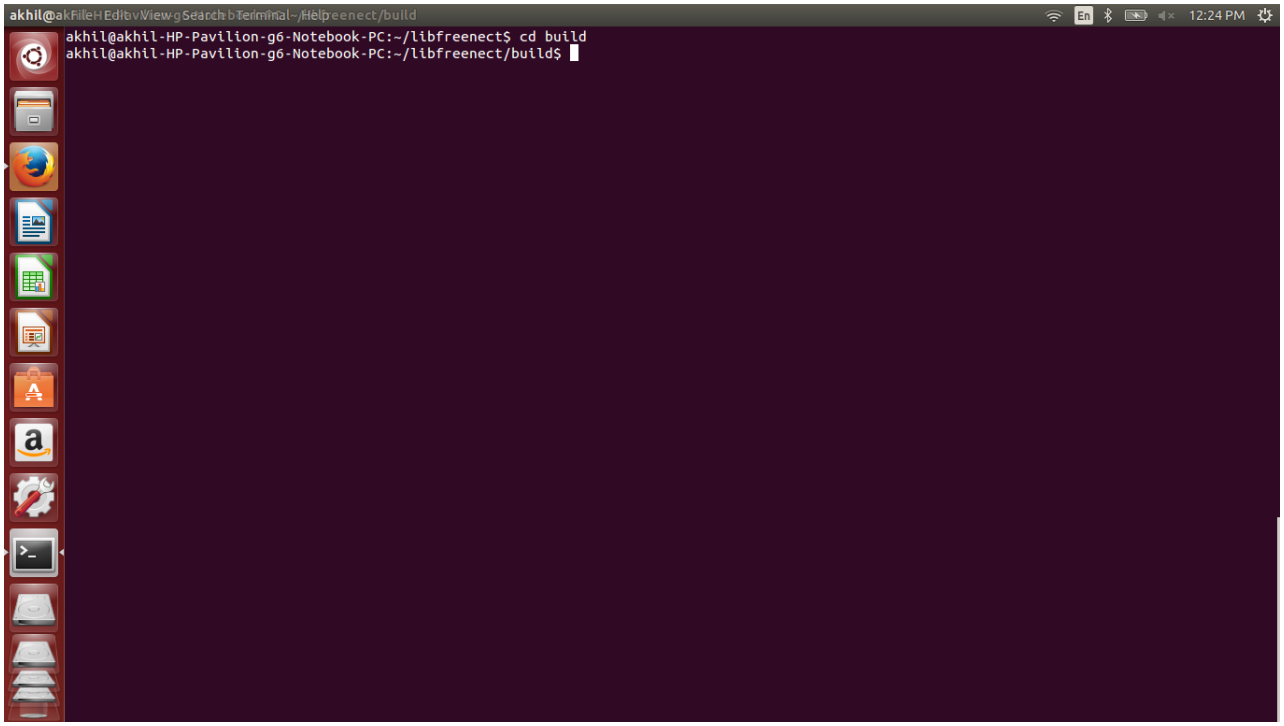


Figure 1.14: Step 5 - Enter the build directory

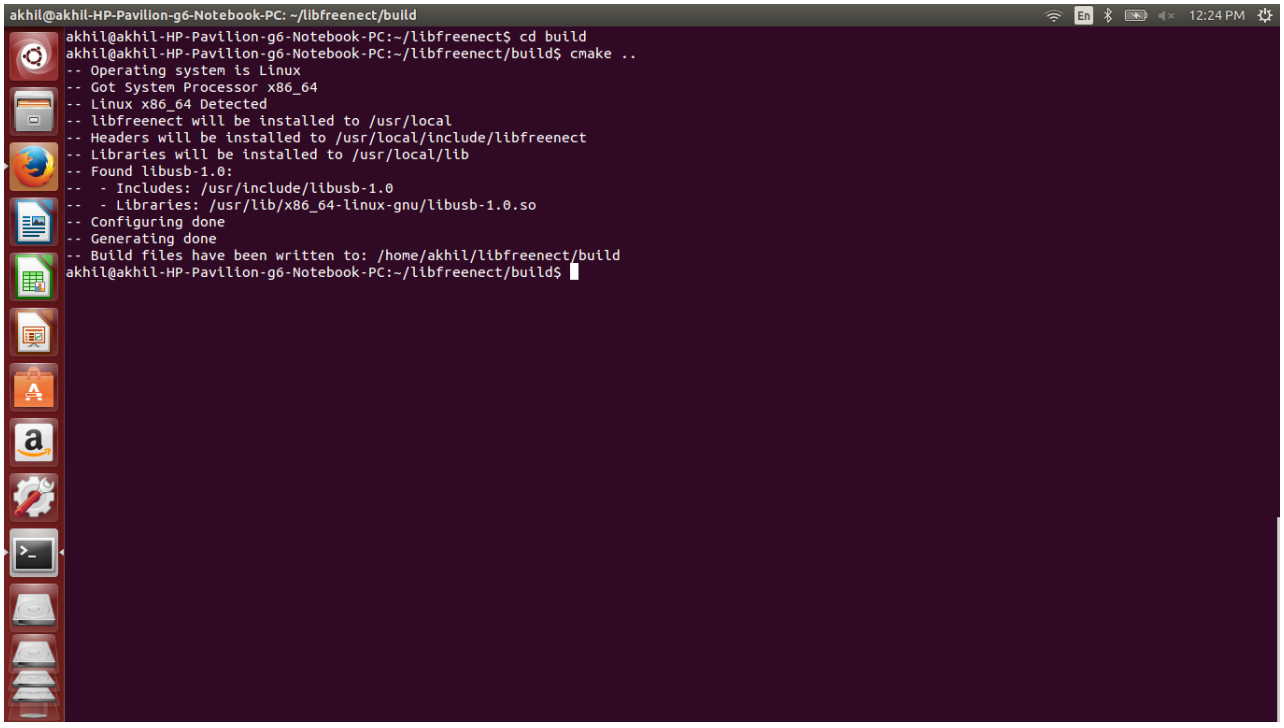


Figure 1.15: Step 6a - cmake ..

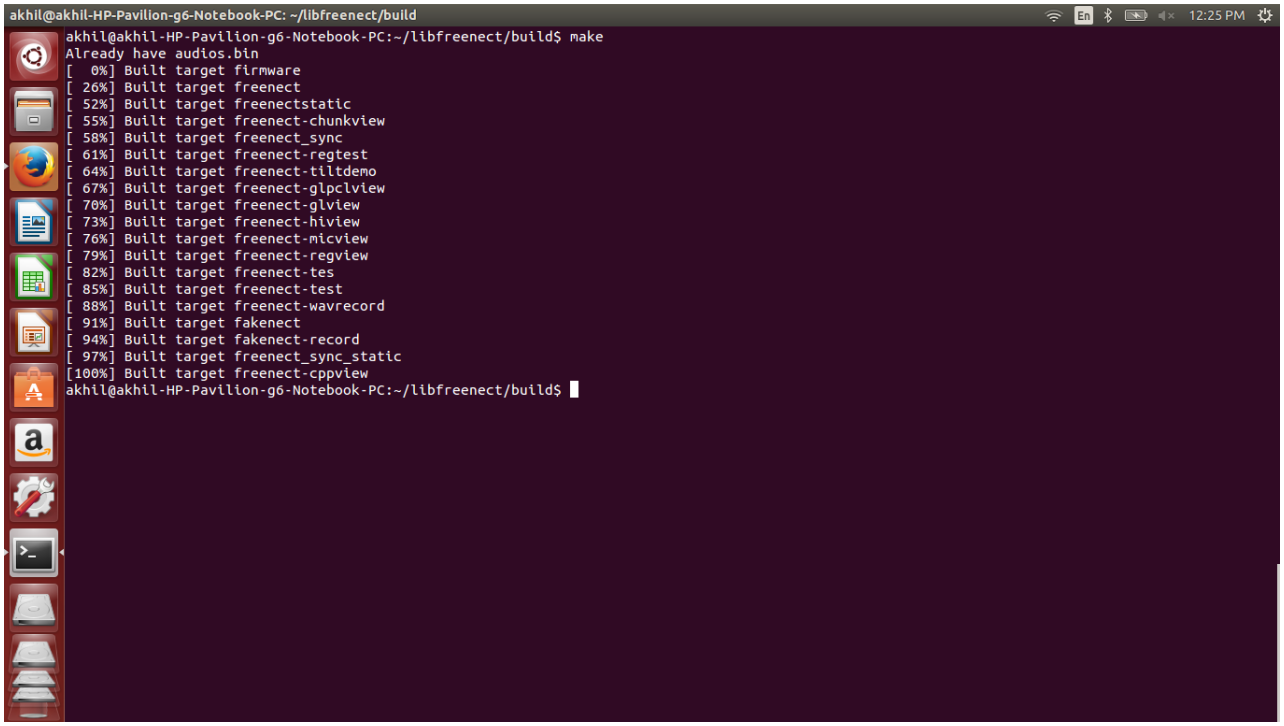


Figure 1.16: Step 6b - make

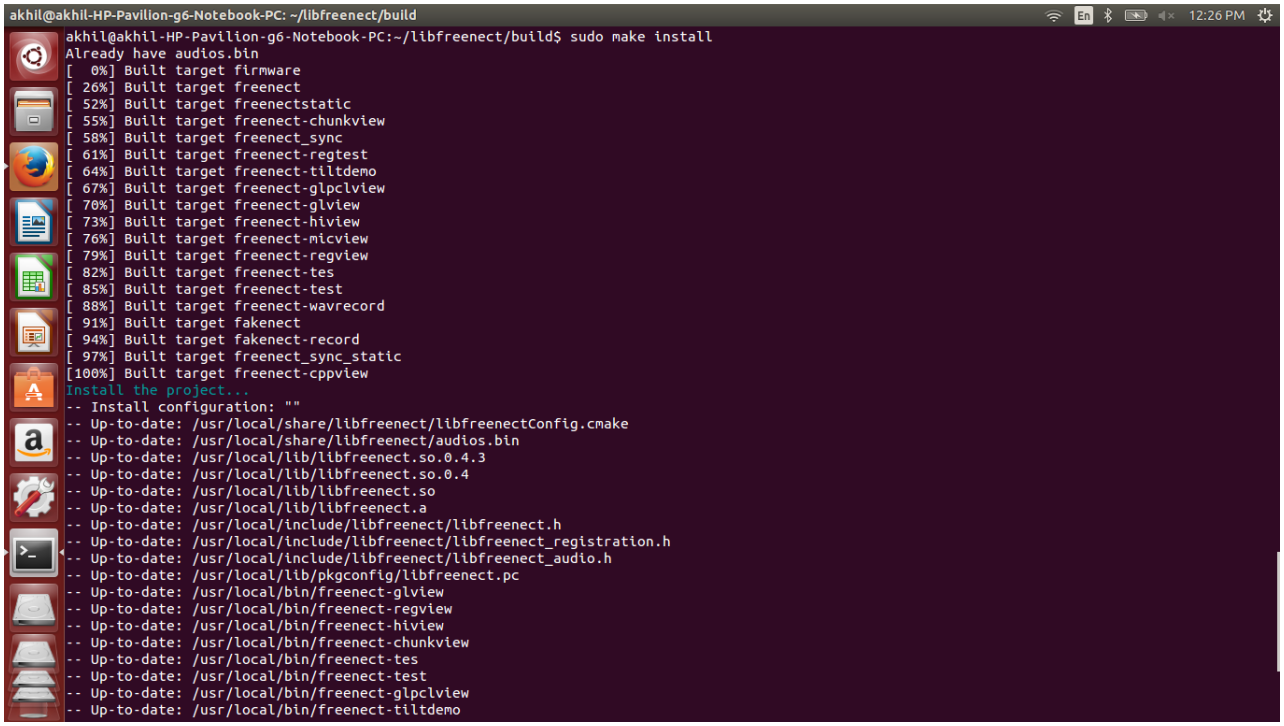


Figure 1.17: Step 7 - sudo make install

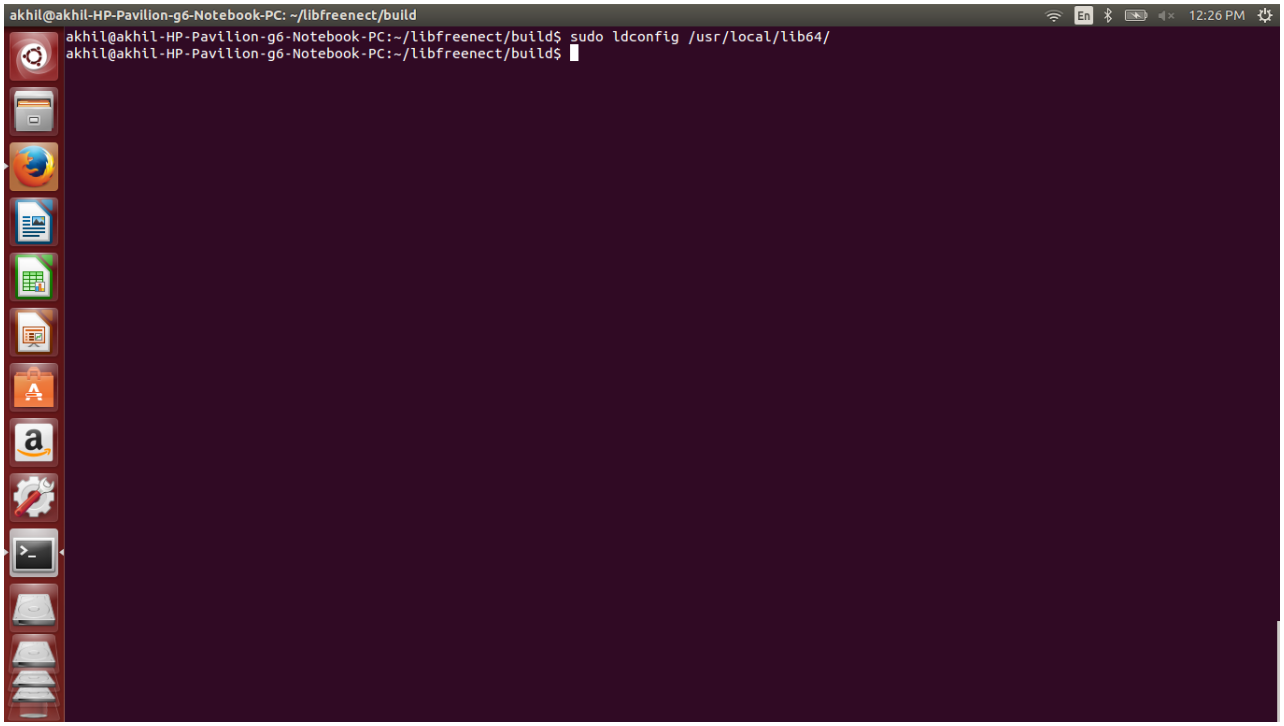


Figure 1.18: Step 8 - Configuring the libraries

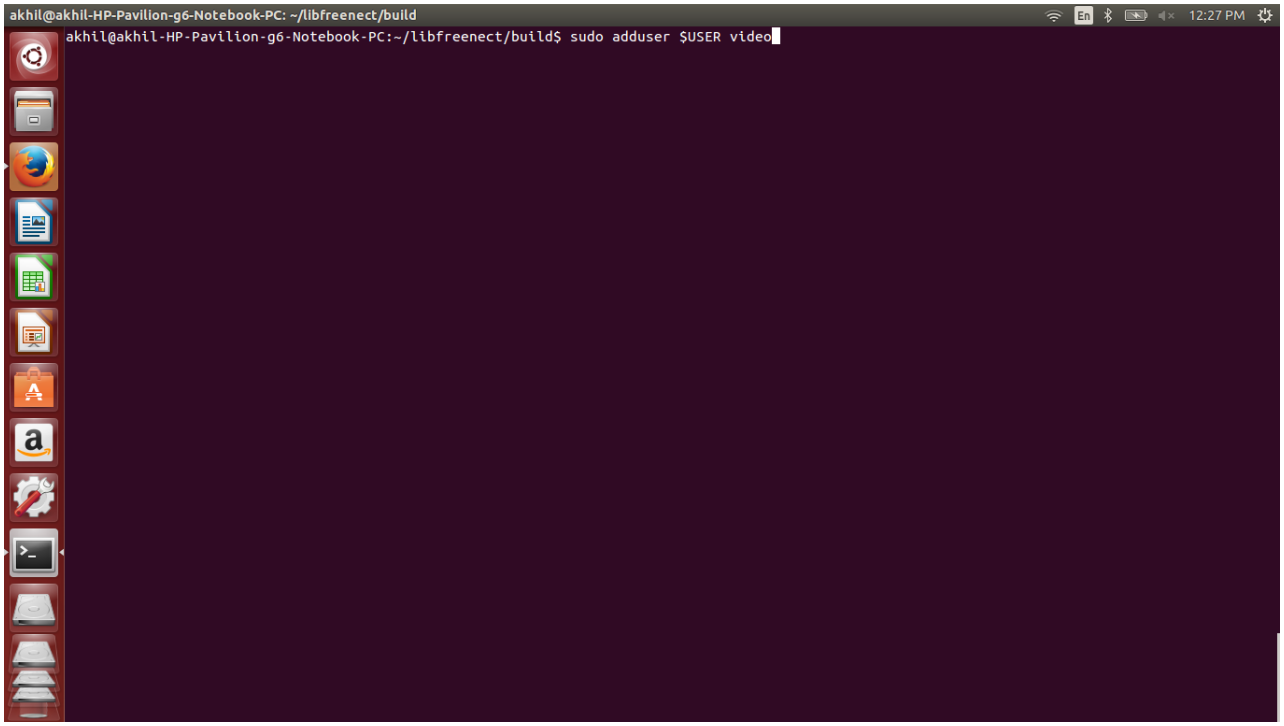


Figure 1.19: Step 9 - To use kinect as a non-root user

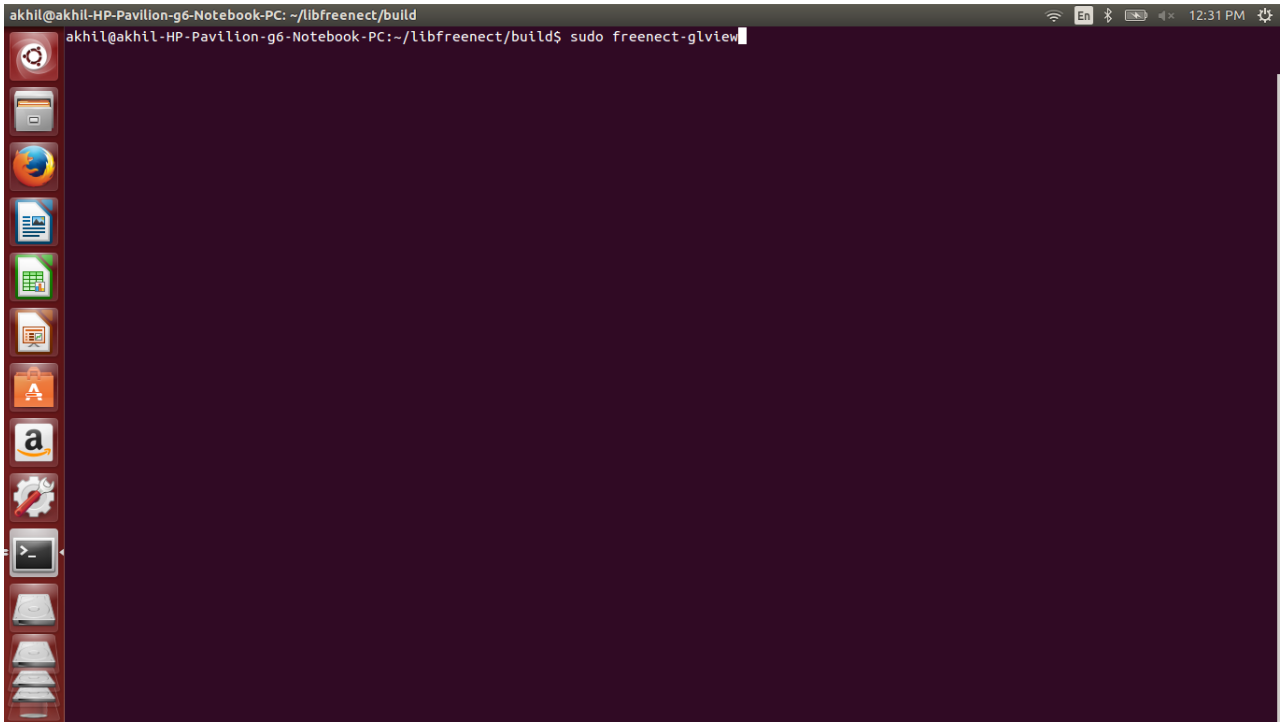


Figure 1.20: Step 10 - Run a sample project

Figure 1.21: Depth Image and Live Stream Video

Chapter 2

Establishing Unicast connection between 2 XBee modules

2.1 Installation of X-CTU (Windows)

Hello world ! This is a quick installation tutorial for Digi's X-CTU software. The X-CTU Software enables you to configure and test various kinds of communication modes between XBee modules.

2.1.1 System Requirements

1. **Supported Operating Systems:** Windows 98, Windows ME, Windows 2K, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 8.1
2. **Non-supported Operating Systems:** Windows 95, Windows NT, Linux
3. **Hardware Requirements:**
   (a) 2 XBee modules
   (b) 1 XBee USB adapter
   (c) Dedicated USB 2.0 bus
   (d) USB A-Type Male to USB B-Type Male Cable
   (e) Firebird V robot

2.1.2 Installation Procedure

Step 1 :

Launch Setup_XCTU_5260.exe to start installation of Digi's X-CTU software for configuration of XBee modules. Click "Next" to continue. Shown in figure 2.1.

Step 2 :

Read the license agreement and select the "I Agree" radio button if you wish to continue.Click "Next". Shown in figure 2.2.

Step 3 :

Select the installation folder you wish to install X-CTU to, and whether you wish to install for Everyone or just the current user.Click "Next". Shown in figure 2.3.

Step 4 :

Confirm the installation by clicking "Next".Shown in figure 2.4.

Step 5 :

Wait till the installation is complete. If you wish to check out Digi's website for firmware updates, click "Yes", else click "No". Shown in figure 2.5.

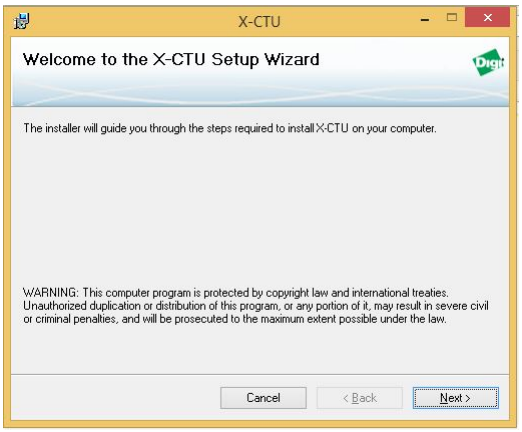


Figure 2.1: Step 1 - Welcome to X-CTU Setup Wizard

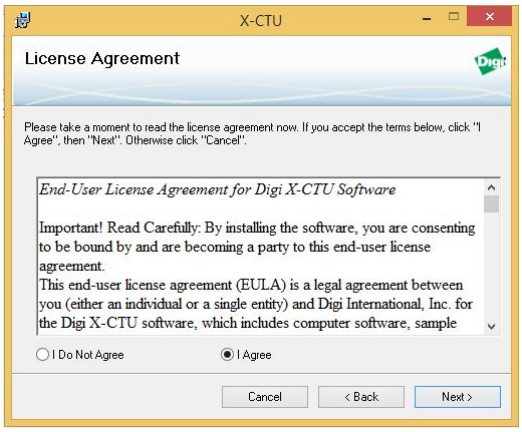


Figure 2.2: Step 2 - License Agreement

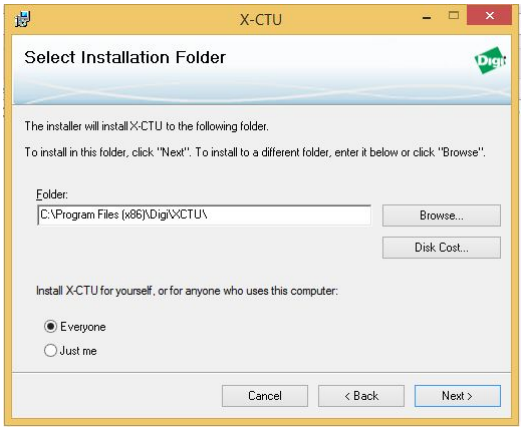


Figure 2.3: Step 3 - Select Installation Folder

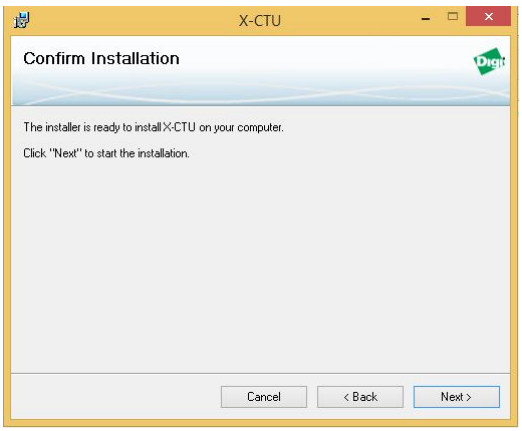


Figure 2.4: Step 4 - Confirm Installation

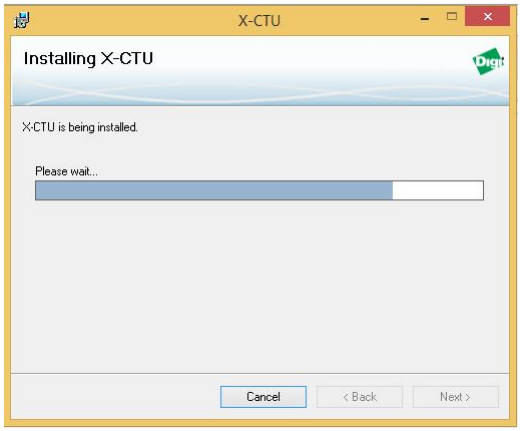


Figure 2.5: Step 5 - Installing X-CTU

Congratulations! X-CTU was successfully installed if you followed the above steps correctly. Click "Close" to close the installation window.

2.1.3 Configuring 2 XBee modules for unicast connection

Hello world ! This is a quick tutorial for configuration of 2 XBee modules for working in Unicast mode.

Step 1 :

Plug in an XBee module in the XBee USB adapter and connect it to a USB port of the PC. Launch X-CTU. Select the "USB Serial Port(COM3)" entry under the "Select Com Port" title. The COM port number varies according to hardware configuration. It might be different for you. Setup the COM port according to the settings shown in the figure. Shown in figures 2.6 and 2.7.

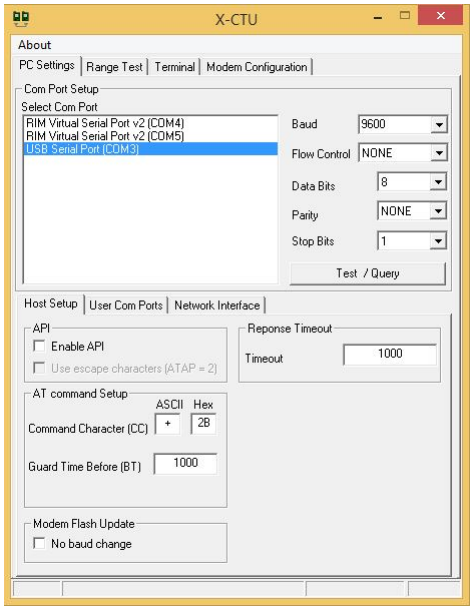


Figure 2.6: Step 1(a) - X-CTU User Interface

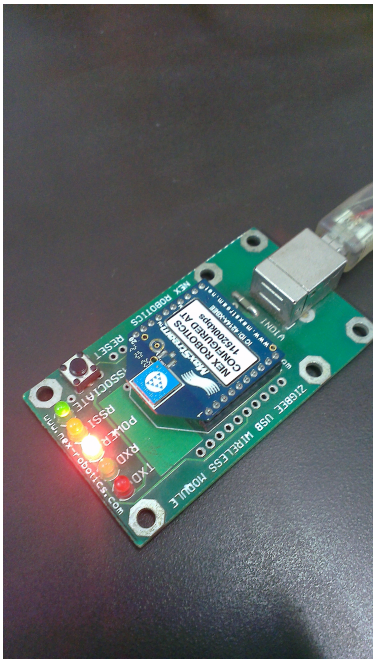


Figure 2.7: Step 1(a) - Connecting XBee module with XBee USB adapter

Step 2 :

Go to the "Modem Configuration" tab and click "Read" under "Modem Parameter and Firmware" section. The Modem Parameters will appear in the viewing area below. If they don't appear and such a pop-up comes up, press the "Reset" button on the Zigbee USB adapter and try reconnecting the adapter and then pressing the "Reset" button if it does not work. Shown in figure 2.8.

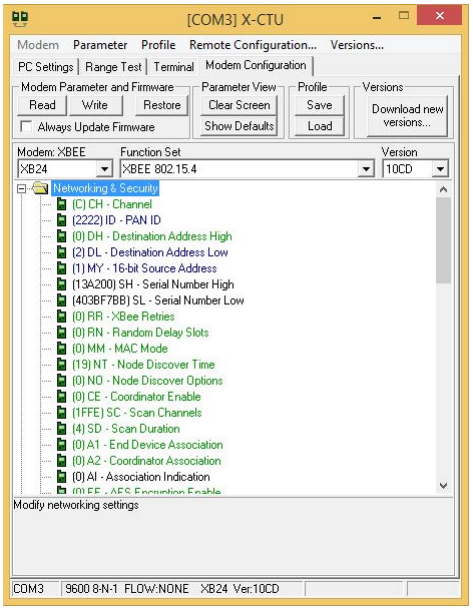


Figure 2.8: Step 2 - Reading the Modem Parameters

Step 3 :

The parameters for both the XBee modules have to be set. Set the Channel and PAN ID of both the XBee modules as same. Set Destination Address High(DH) for both the modules as 0. The Destination Address Low(DL) and 16-bit Source Address(MY) have to be set such that the DL of one module is the MY of the other and MY of one module is the DL of other for both the modules. Here DL=2, MY=1 was used for the first module, and vice-versa for the other.

Configuration of the 2 Zigbee modules can be done one after another or simultaneously by launching 2 instances of X-CTU and connecting both the modules with the PC. Following these steps will configure the X-Bee modules in unicast mode, i.e. simple one-to-one wireless communication. It is recommended not to change the rest of the parameters and leave them at their default values. However, if you are an advanced user, you may tinker with their values if you know what you are doing. Shown in figures 2.9 and 2.10 for both the modules.

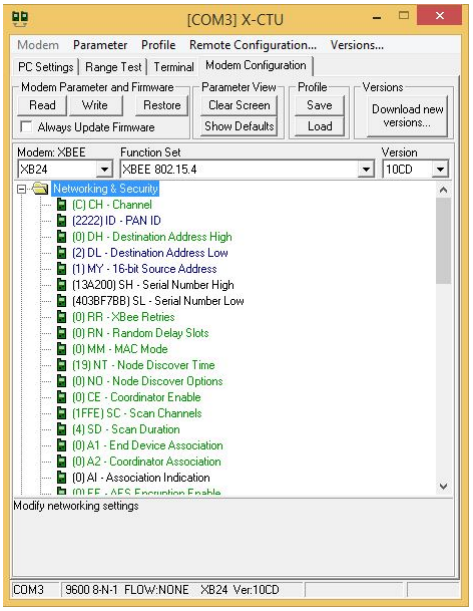


Figure 2.9: Step 3(a) - Configuring the Modem parameters for XBee module 1

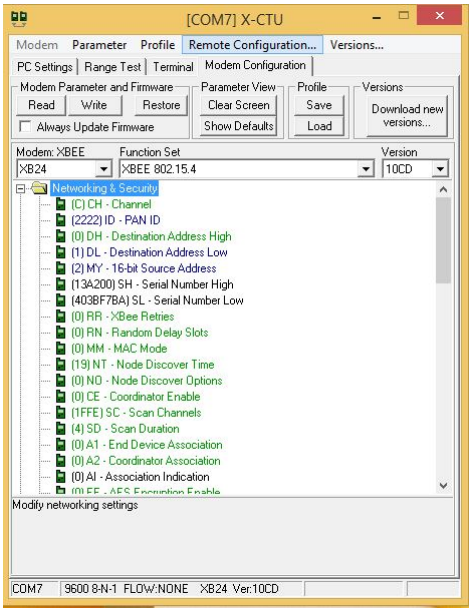


Figure 2.10: Step 3(b) - Configuring the Modem parameters for XBee module 2

Step 4 :
After configuration of both the XBee modules, leave one of the modules in the USB Adapter and plug the other one in the Firebird V robot's X-Bee slot. Load the program "_Interfacing_Kinect_To_Firebird_Using_Zigbee.hex" provided with this document on the Firebird V robot. Start the X-CTU software and go to the "Terminal" tab. You can send the byte sized commands shown in Table 2.1 to the Firebird V robot to test the wireless connection via XBee:

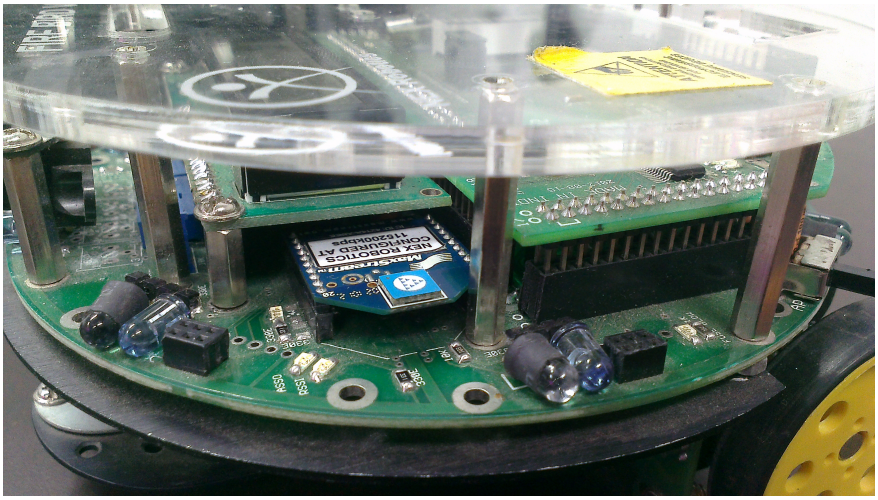


Figure 2.11: Step 4(a) - Connecting Zigbee module to Firebird V

Figure 2.12: Step 4(b) - X-CTU's terminal Wizard

Congratulations! If you followed the above steps correctly, you just established a unicast connection between 2 X-Bee modules. Shown in figure 2.11 and 2.12.

Keyboard Key	ASCII Value	Action
8	0x38	Forward
2	0x32	Backward
4	0x34	Left
6	0x36	Right
5	0x35	Stop
7	0x37	Buzzer On
9	0x39	Buzzer Off

Table 2.1: Firebird V Commands

Chapter 3

Set up Process on Visual Studio

3.1 Introduction

Visual Studio Express products provide a free development environment to develop applications for the latest platforms. Visual Studio Express can be downloaded from

http://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx.

The experiments on kinect discussed in this manual are designed using C# and Windows Presentation Foundation (WPF). Before commencing the discussion of experiments and their code, it is essential to understand how to set up the environment to design the same. This includes adding references, the procedure for which is discussed in this chapter. This is a prerequisite to the experimenting stage to follow.

Note : At the time of making of this manual, the latest version was Visual Studio Express 2013. The same was used for experimenting with kinect.

3.1.1 System Requirements

Supported Operating System
Windows 7 SP1 (x86 and x64), Windows 8 (x86 and x64), Windows 8.1 (x86 and x64), Windows Server 2008 R2 SP1 (x64), Windows Server 2012 (x64) and Windows Server 2012 R2 (x64).

- Hardware Requirements
  - 1.6 GHz or faster processor
  - 1 GB of RAM (1.5 GB if running on a virtual machine)
  - 5 GB of available hard disk space
  - 5400 RPM hard drive
  - DirectX 9-capable video card running at 1024 x 768 or higher display resolution

3.1.2 Set up Procedure

Step 1 :

Double click the **VS Express 2013 for desktop** icon to start the application. Alternately, one can start the application from start menu as well. Shown in Figure 3.1.

Step 2 :

Click on **New Project** to create a new project. Shown in Figure 3.2.

Step 3 :

Select **Visual C#** → **WPF Application** Give a project name and save it in the appropriate directory of your choice. In this example the project name is **my_expt** saved in the **experiment** folder. Shown in Figure 3.3.

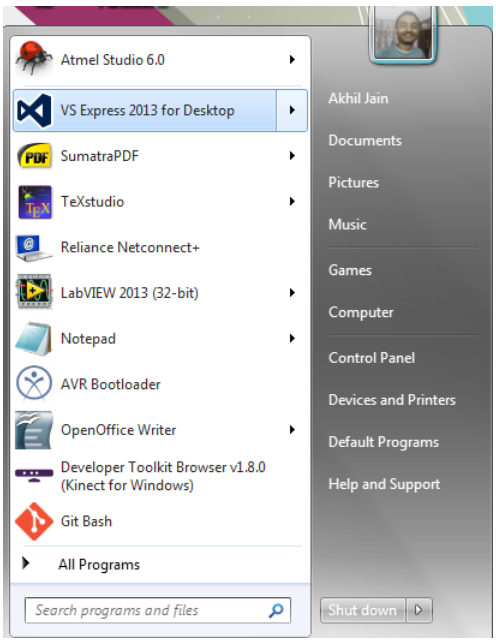


Figure 3.1: Step 1 - Open VS Express 2013 for windows desktop

Step 4 :

The project window will open. On the right hand side you will find the **Solution Explorer**. Right click on **my_expt** → **Add** → **Reference** Shown in Figure 3.4.

Step 5 :

The Reference Manager will open. Go to **Assemblies** → **Extensions** and select **Microsoft.Kinect**. Click 'OK' to add. Shown in Figure 3.5.

Step 6 :

In the solution explorer right click on **Solution 'my_expt'(1 project)** → **Add** → **Existing Project** Shown in Figure 3.6.

Step 7 :

Navigate to **KinectWpfViewers** folder (provided with this manual) and select **Microsoft.Samples.Kinect.WpfViewers.csproj**. Click 'Open' to add this project. Shown in Figure 3.7.

Step 8 :

Now we need to reference this project. Right click on **my_expt** → **Add** → **Reference**. Go to **Solution** → **Projects** and select **Microsoft.Samples.kinect.WpfViewers** Shown in Figure 3.8.

Step 9 :

Now add the final 2 references. Right click on **my_expt** → **Add** → **Reference**. Go to Browse and navigate to **KinectWpfViewers** folder (provided with this manual) → **bin** → **Debug** and select the 2 files : **Microsoft.Kinect.Toolkit.dll** and **Microsoft.Samples.Kinect.WpfViewers.dll** Click 'OK' to add the same.
Shown in Figure 3.9.

Step 10 :

This completes the set up process. The Project window after completion of setup is shown in Figure 3.10. You can now start programming on C# and WPF for kinect applications

Congratulations ! You have successfuly completed the set up on Visual Studio to design codes for working on the Kinect sensor.

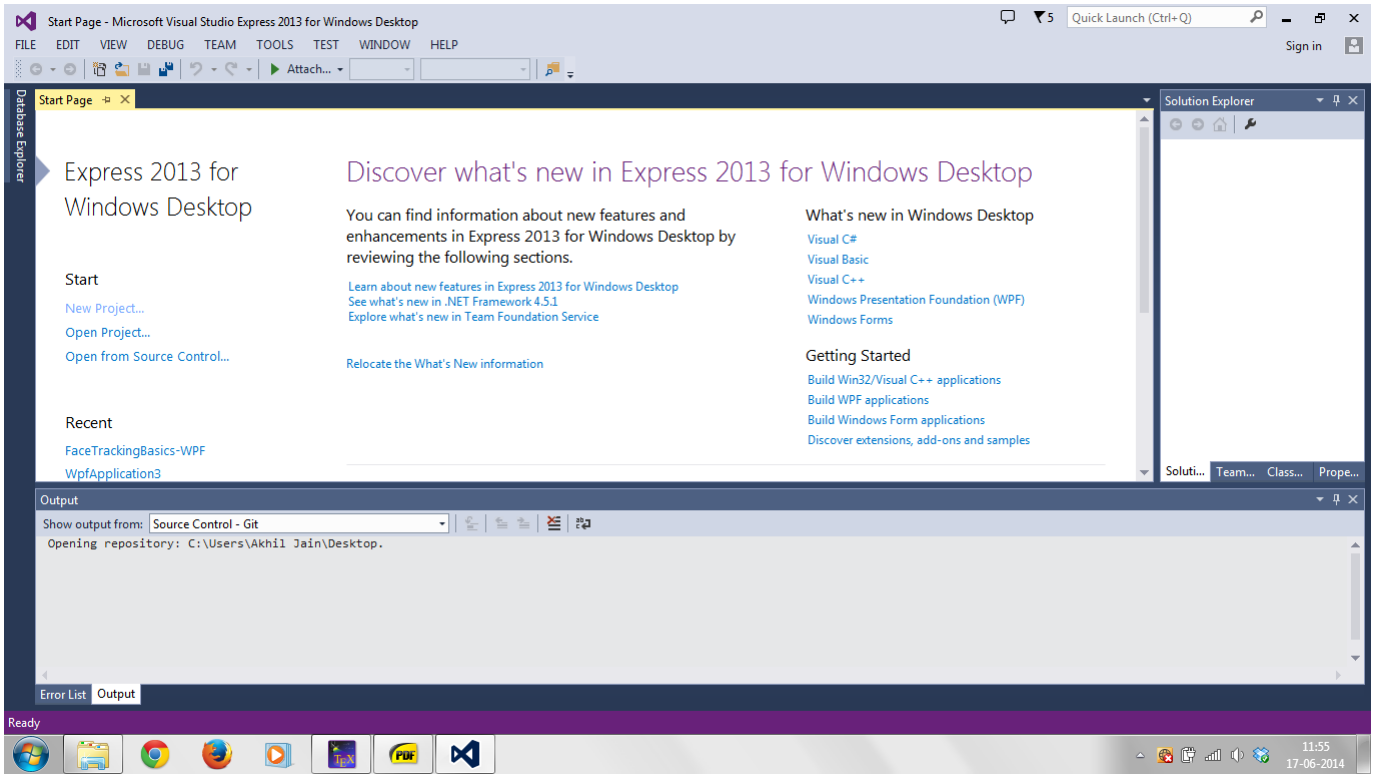


Figure 3.2: Step 2 - Click on New Project

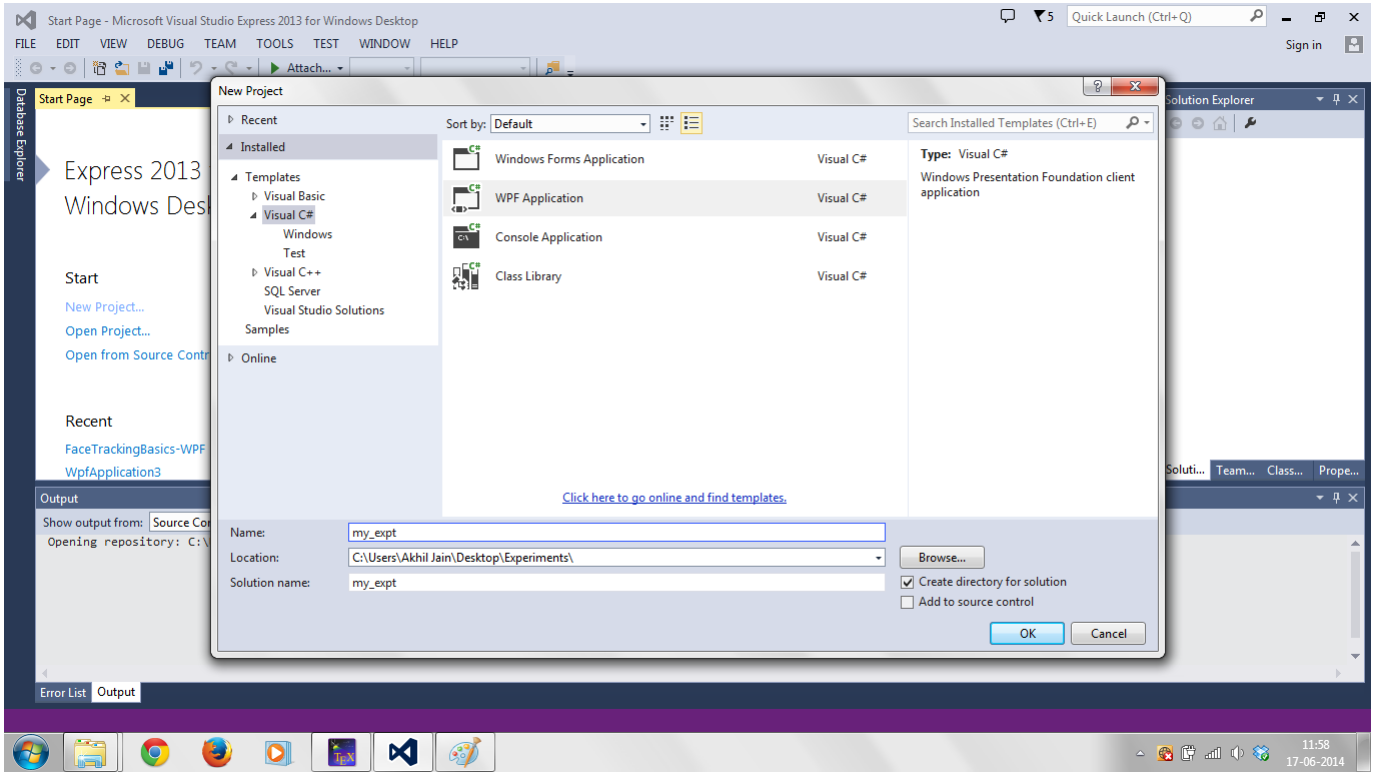


Figure 3.3: Step 3 - Create a new C# project

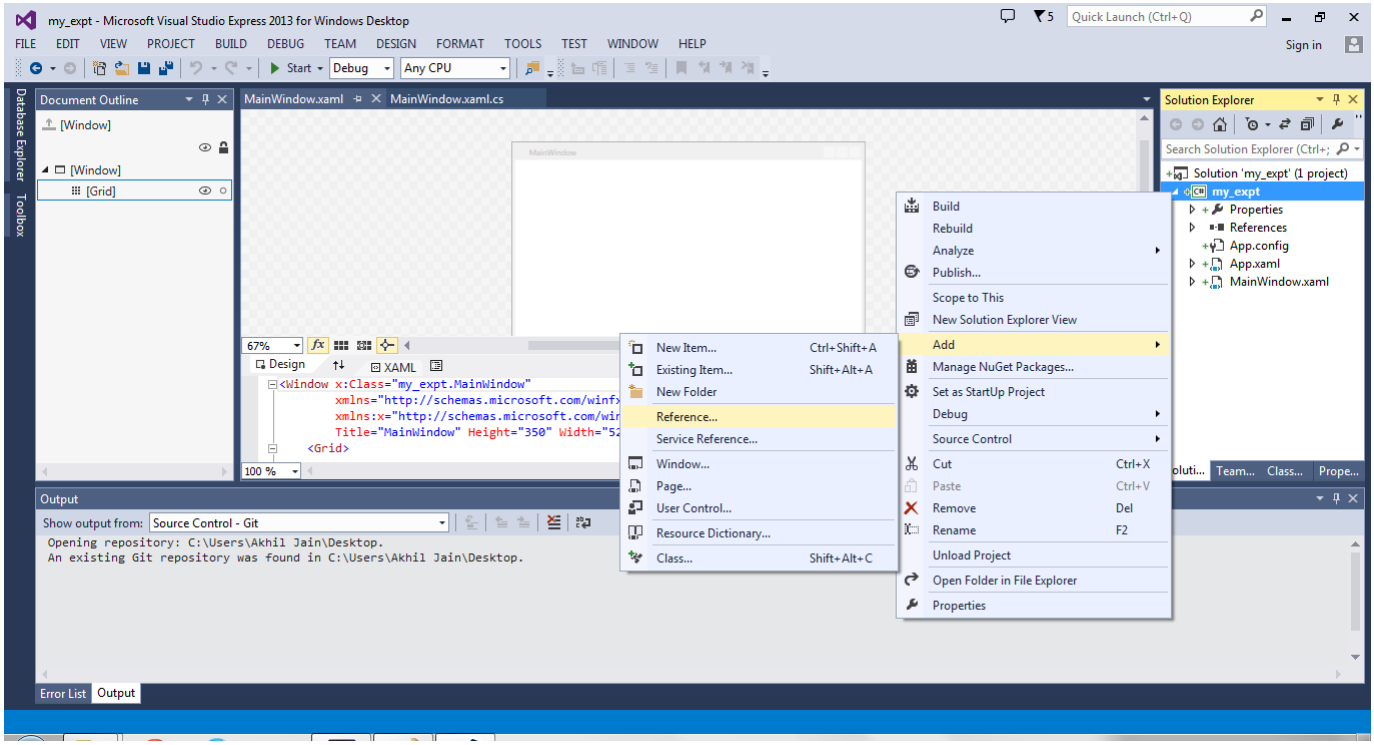


Figure 3.4: Step 4 - Adding Reference

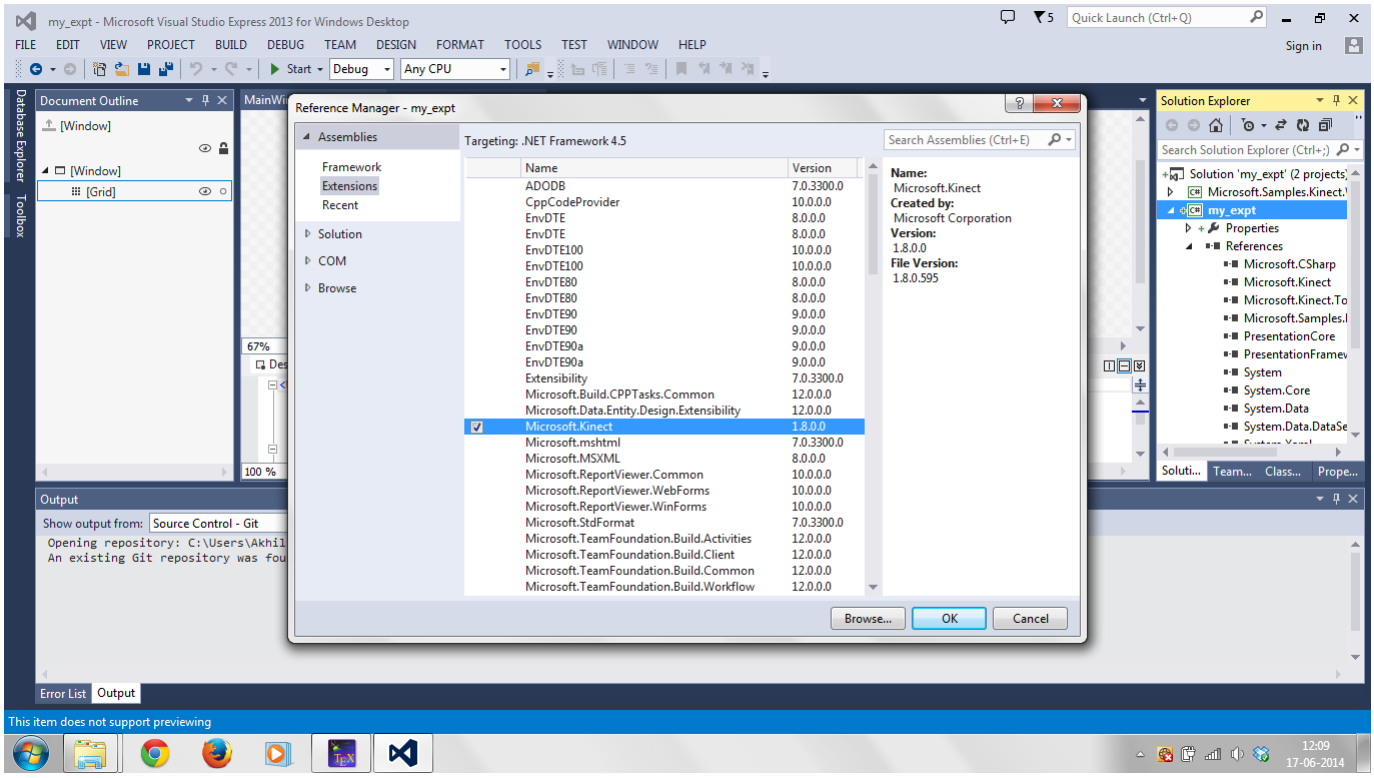


Figure 3.5: Step 5 - Add reference to Microsoft.Kinect

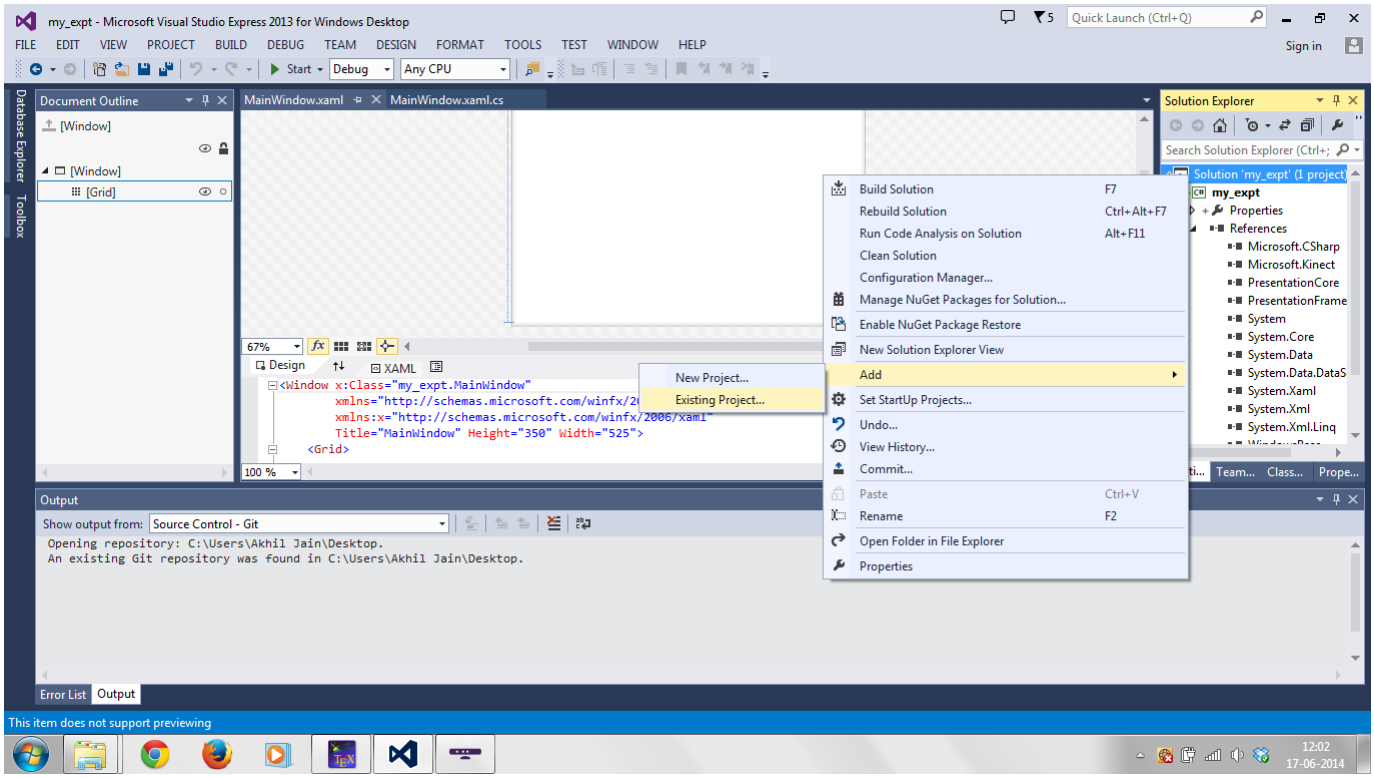


Figure 3.6: Step 6 - Adding existing project

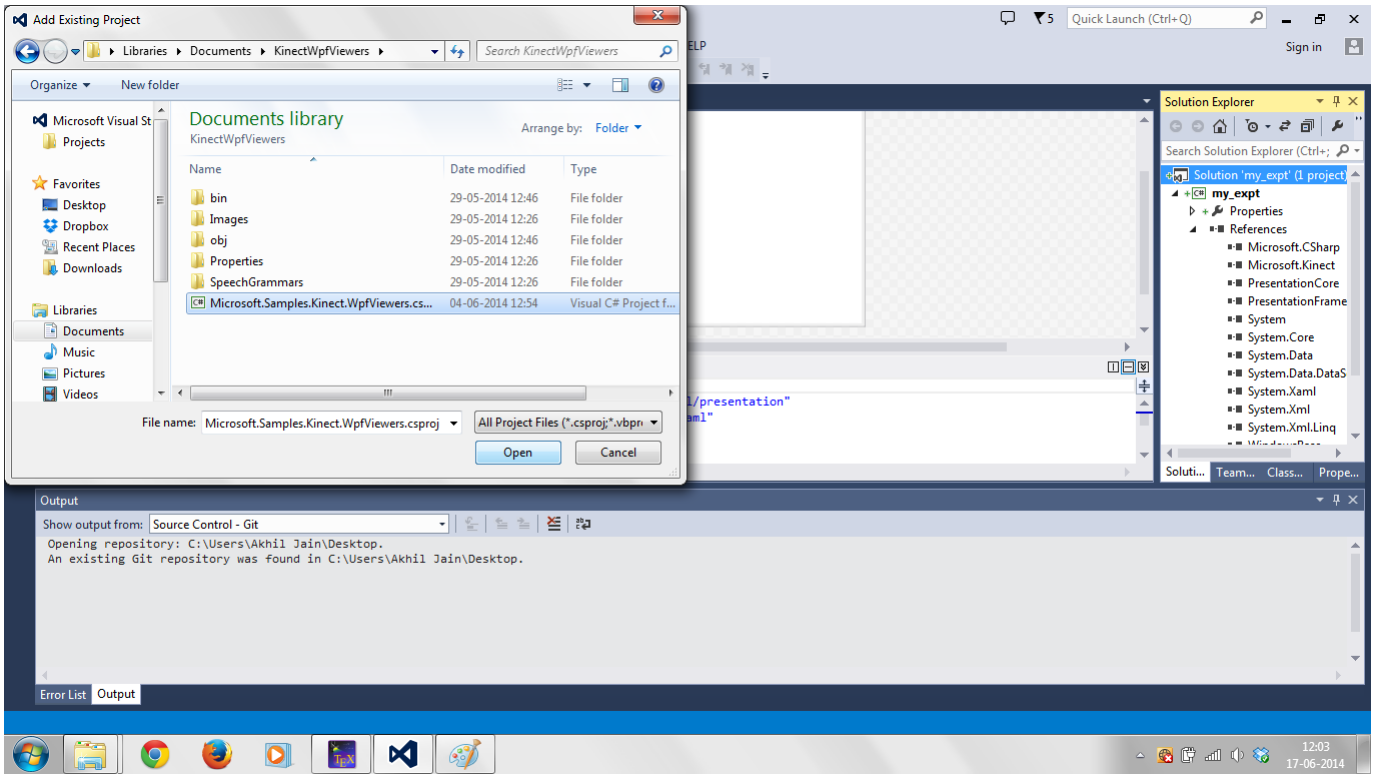


Figure 3.7: Step 7 - Select the project in KinectWpfViewers

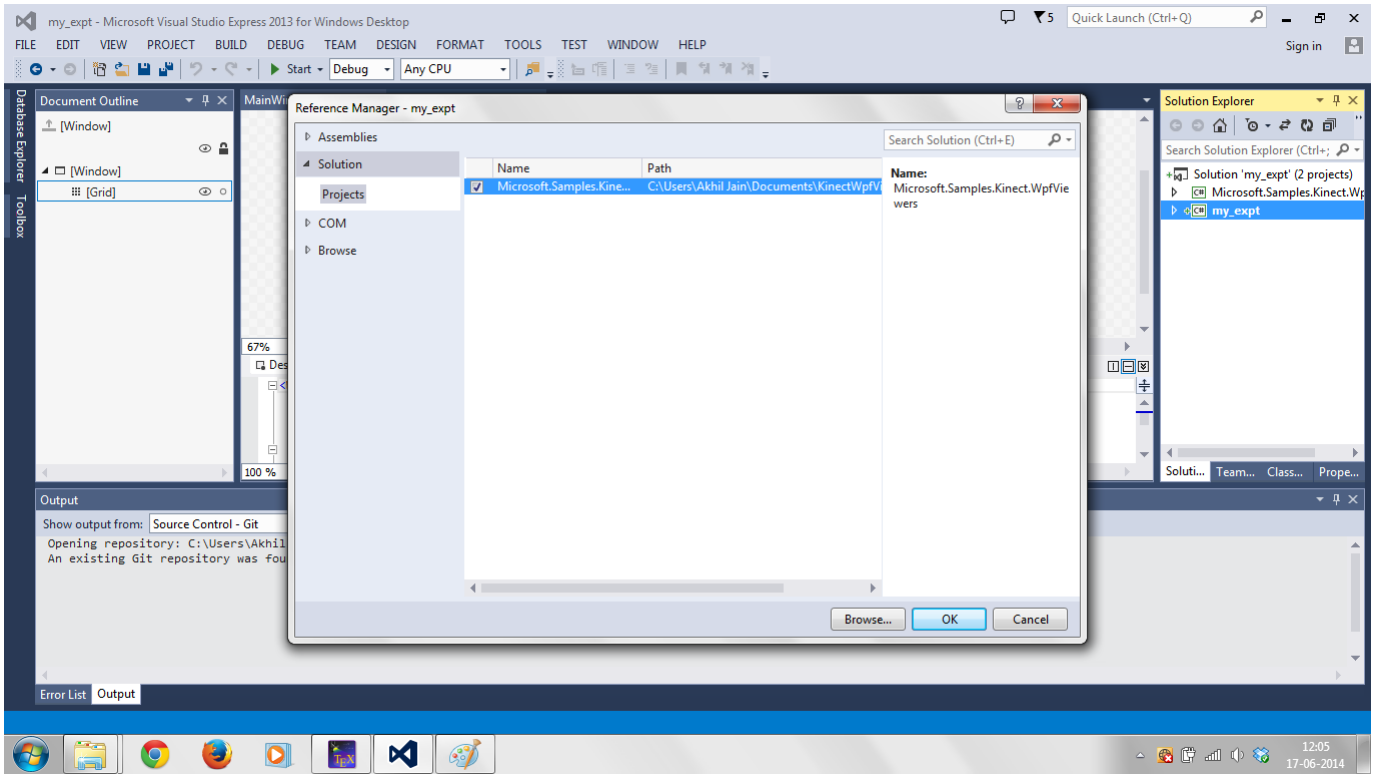


Figure 3.8: Step 8 - Reference the added project

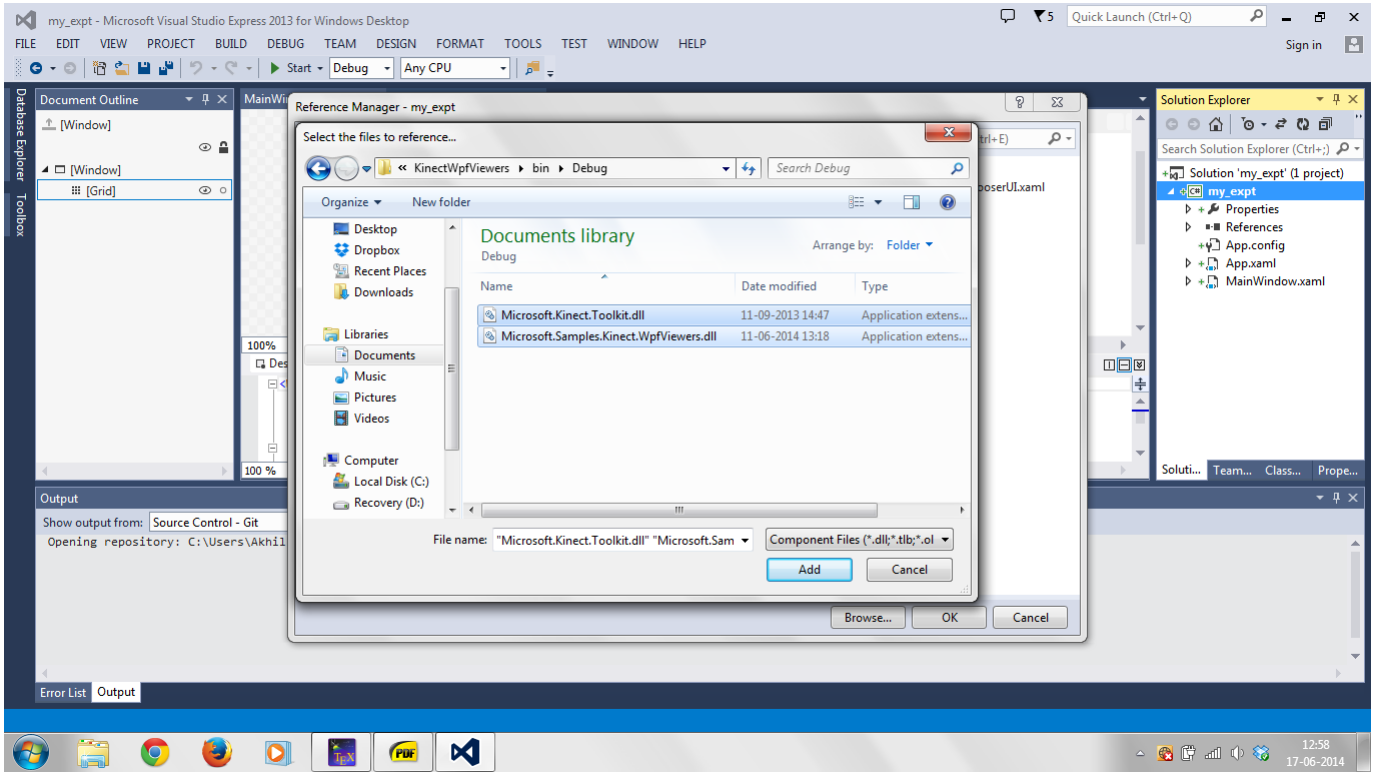


Figure 3.9: Step 9 - Adding final 2 references

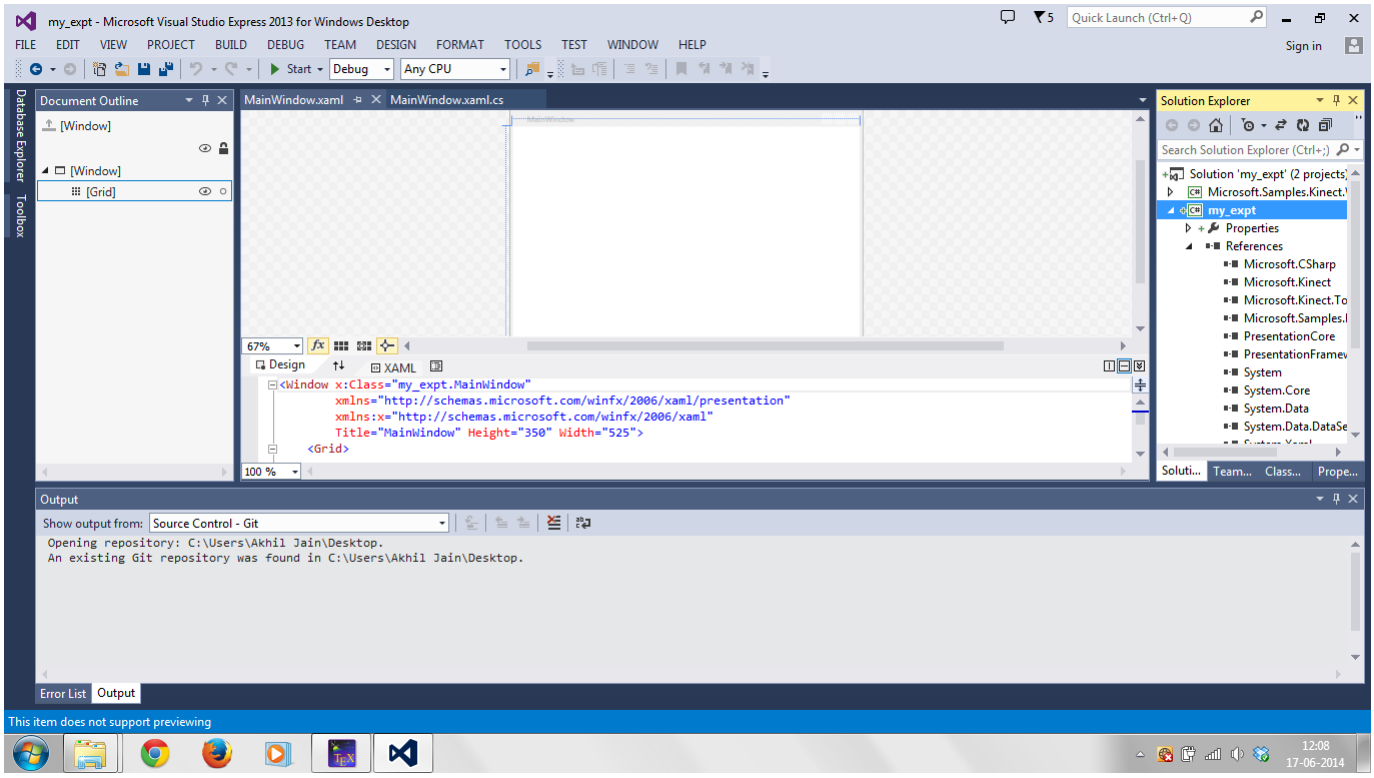


Figure 3.10: Step 10 - Project window post setup

Chapter 4

Experiments - The fun part

At this stage you must have installed the necessary software required to work on the kinect sensor on your platform. So without further delay lets begin the experiments with kinect.

The following sections showcase the experiments designed on the windows platform :
4.1 Depth tracking on screen
4.2 Depth tracking with firbird
4.3 Camera fundamentals
4.4 Skeleton tracking fundamentals
4.5 Skeleton tracking with firebird (left and right)
4.6 Skeleton tracking with firebird (front and back)
4.7 Skeleton tracking (angle between joints)
4.8 Skeleton tracking (Hover Button)
4.9 Skeleton tracking complete
4.10 Voice recognition
4.11 Tilt Demo

The following sections showcase the experiments were designed on the linux platform :
4.12 Depth tracking on screen
4.13 Depth tracking with firebird
4.14 Tilt demo

In the following pages the experiments will be discussed in detail.
Refer to [4] [5] [6] [7] [8] [9] [10] [11].

4.1 Depth tracking on screen

4.1.1 Aim of the experiment

The aim of this experiment is to obtain the depth information of every pixel in the frame captured by the kinect.

4.1.2 Components required

- Kinect sensor.

4.1.3 Experiment description

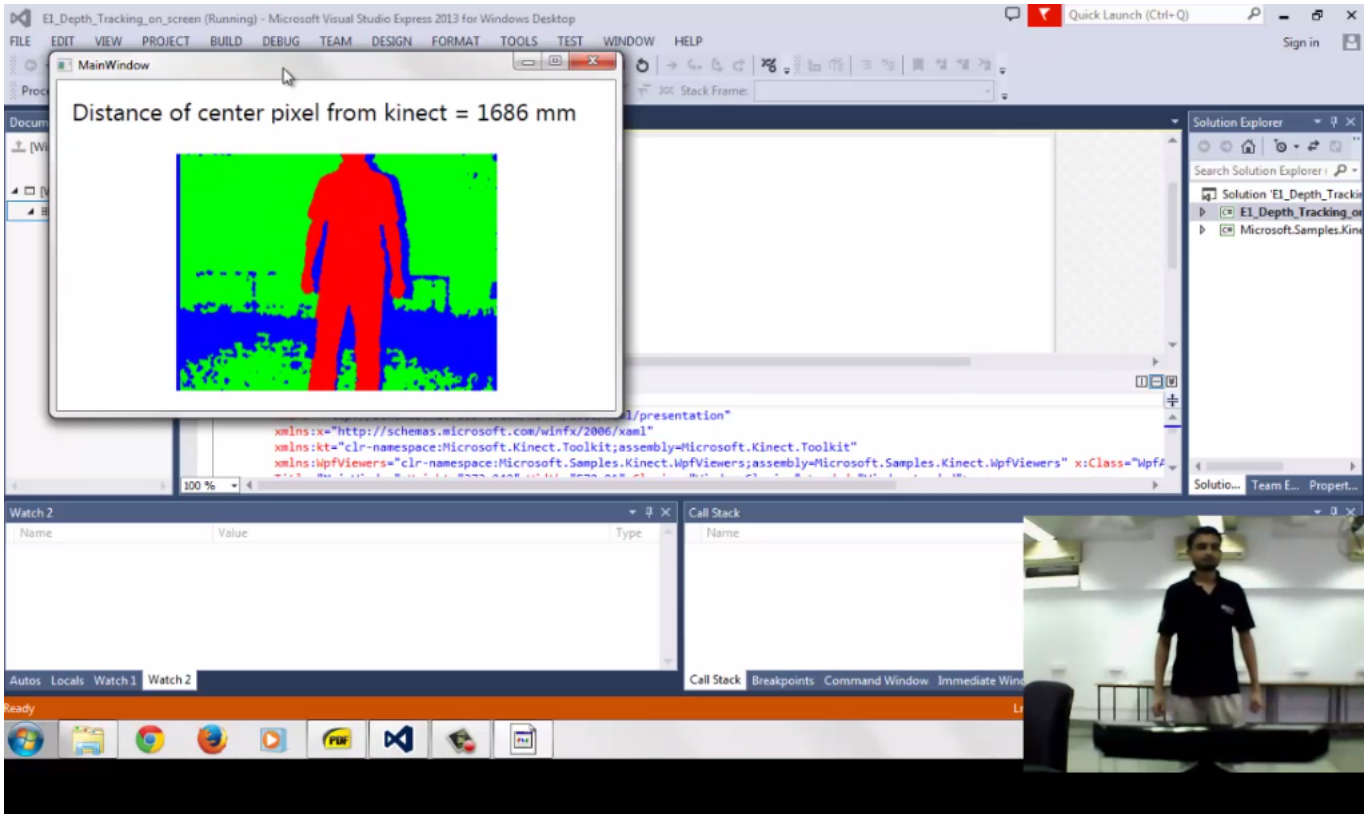
This experiment obtains the depth value of every pixel in the frame. Based on this obtained information a color coding scheme is designed. If the pixel is at a distance of less than 900mm (and greater than 800mm) it is assigned the color Blue. If the pixel is at a distance greater than 900mm and less than 200mm the color assigned to it is Green. If the pixel is at a distance greater than 2000mm then it is assigned the color Red. Thus the output frame comprises of Blue Green or Red colored pixels depending upon its distance from the kinect. The distance of the center pixel of the frame is displayed on the screen for simplicity and validity of the experiment.

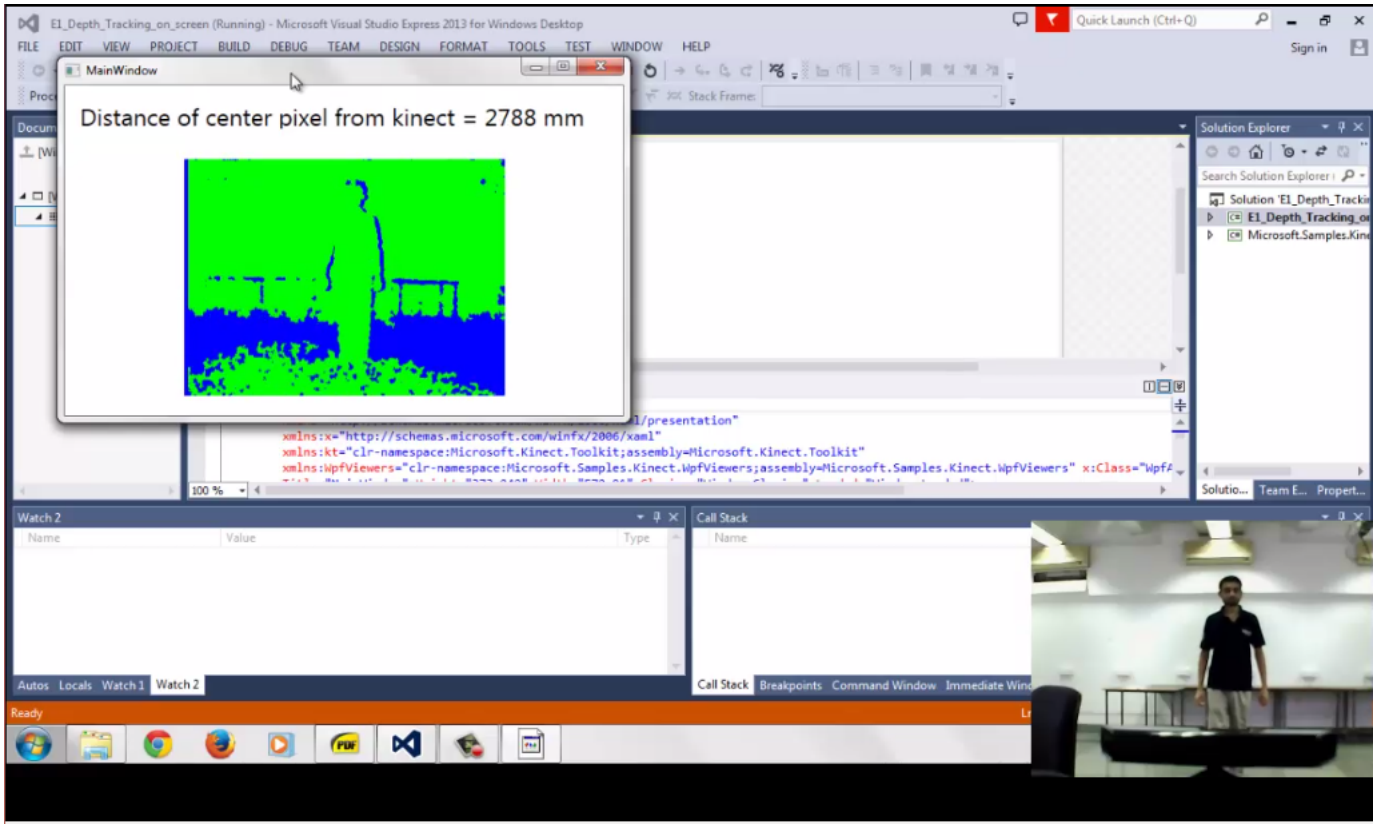
The experiment is available in the folder **E1_Depth_Tracking_on_screen** provided.

4.1.4 Instructions

1. Click on the 'start' button to run the program.
2. Make sure that the background doesnot contain windows because sunlight (or any source of light) is interpreted being within the 900mm range. The violation of this step, however, will not affect the functionality of this experiment.
3. The user must stand at a distance of atleast 800mm from the kinect. This is the lower threshold of the kinect itself, to get depth data.
4. When done experimenting either close the main window or click on the 'Stop' button to stop the execution of the code.

4.1.5 Experiment Outputs





4.1.6 Conclusion

The experiment was successfully conducted to obtain the depth information of every pixel in the frame captured by the kinect and the depth value of the center pixel was obvserved.

4.2 Depth tracking on firebird

4.2.1 Aim of the experiment

The aim of this experiment is to obtain the depth information of every pixel in the frame captured by the kinect and send appropriate commands to the firebird 5 robot based on it.

4.2.2 Components required

- Kinect sensor
- Firebird V Robot
- Zigbee Modules (2 nos.)
- Zigbee USB adapter

4.2.3 Experiment description

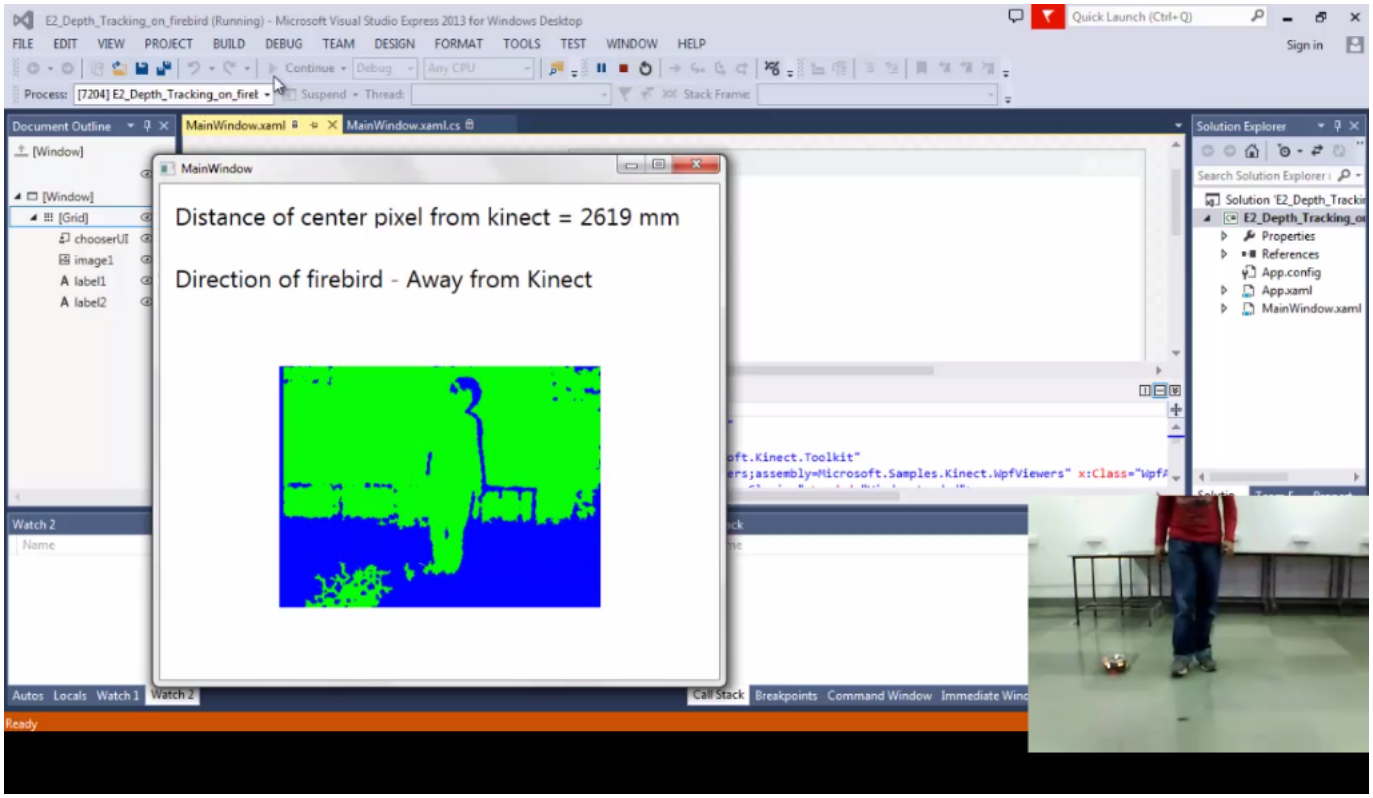
This experiment obtains the depth value of every pixel in the frame. Based on this obtained information a color coding scheme is designed. If the pixel is at a distance of less than 900mm (and greater than 800mm) it is assigned the color Blue. If the pixel is at a distance greater than 900mm and less than 200mm the color assigned to it is Green. If the pixel is at a distance greater than 2000mm then it is assigned the color Red. Thus the output frame comprises of Blue Green or Red colored pixels depending upon its distance from the kinect. The distance of the center pixel of the frame is displayed on the screen. If the user moves towards the kinect, the command "58" is sent through the zigbee module, '5' for 'stop' and '8' for 'forward'. If the user moves away from the kinect, the command "52" is sent through the zigbee module, '55' for 'stop' and '2' for 'reverse'.

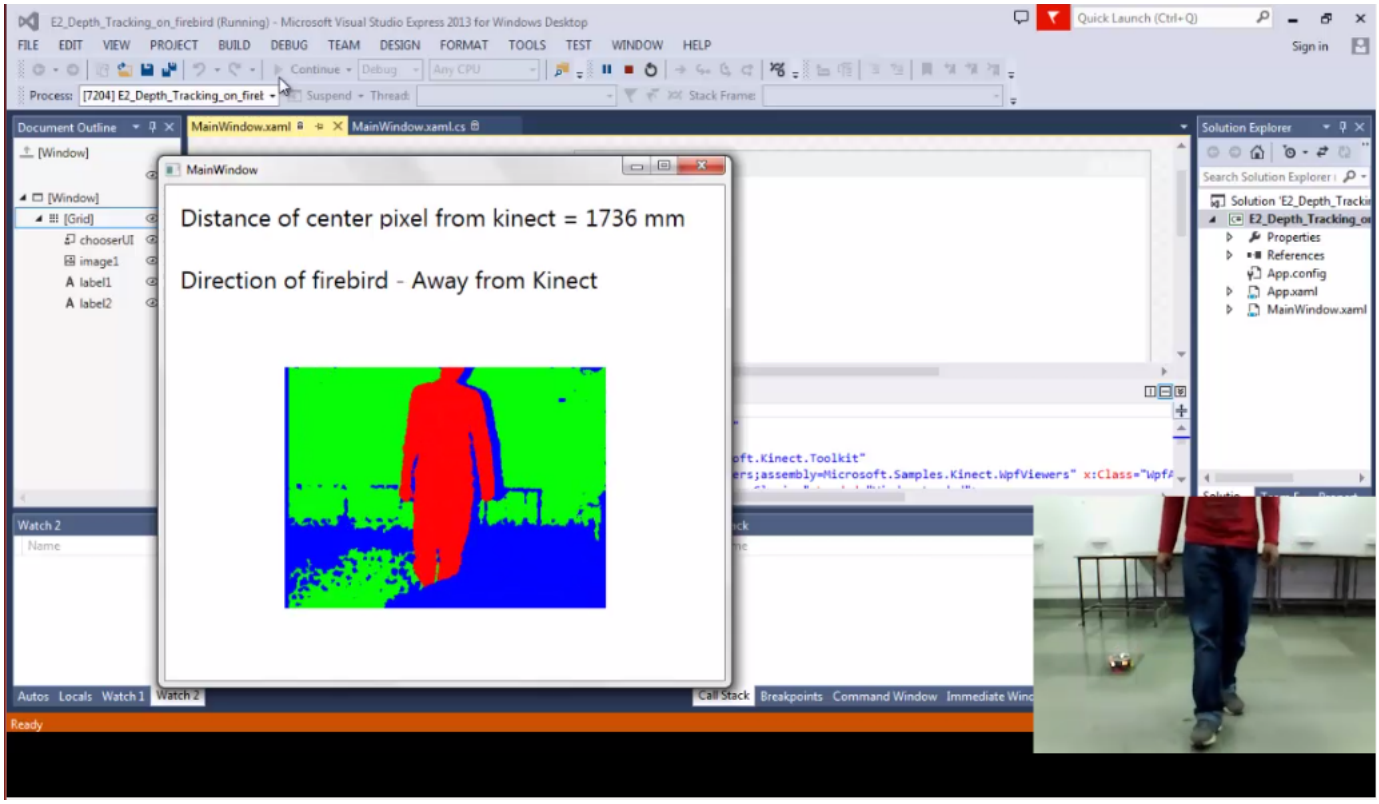
The experiment is available in the folder **E2_Depth_Tracking_on_firebird** provided.

4.2.4 Instructions

1. Click on the 'start' button to run the program.
2. Turn on the firebird 5 robot loaded with the zigbee serial interfacing code and establish unicast connection between two zigbee modules as described in the manual.
3. Make sure that the background doesnot contain windows because sunlight (or any source of light) is interpreted being within the 900mm range. The violation of this step, however, will not affect the functionality of this experiment.
4. The user must stand at a distance of atleast 800mm from the kinect. This is the lower threshold of the kinect itself, to get depth data.
5. Move towards the kinect to make the firebird move forward and away from the kinect to make it move in reverse.
6. When done experimenting either close the main window or click on the 'Halt' button to stop the execution of the code.

4.2.5 Experiment Outputs





4.2.6 Conclusion

- The experiment was successfully conducted to obtain the depth information of every pixel in the frame captured by the kinect.
- The depth value of the center pixel was obvserved, and the appropriate commands were sent to the firebird V robot based on it.

4.3 Camera Fundamentals

4.3.1 Aim of the experiment

The aim of this experiment is to obtain the live stream of the kinect camera on screen in 320x240 resolution.

4.3.2 Components required

- Kinect sensor

4.3.3 Experiment description

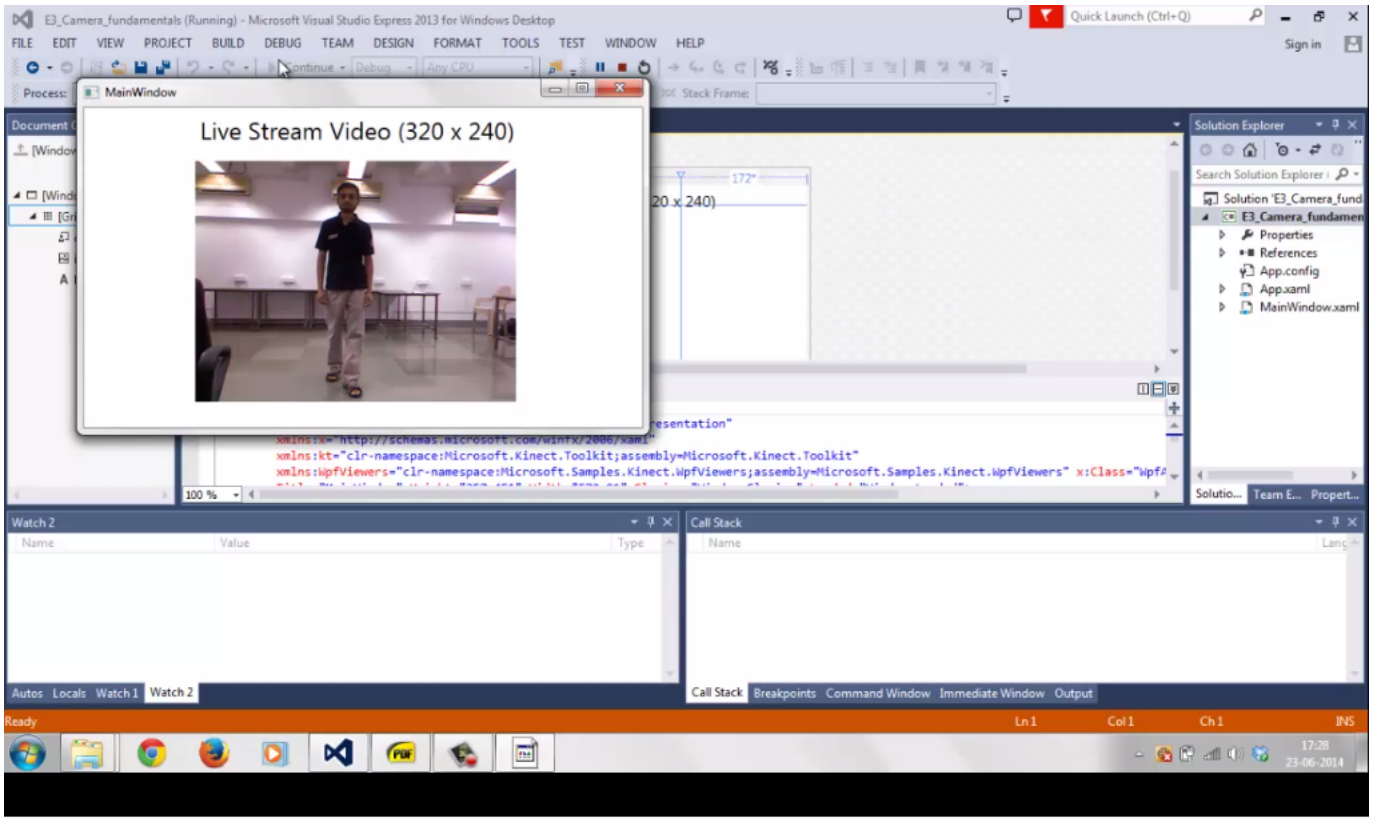
This experiment obtains the live video stream from the kinect camera and display it on the screen.

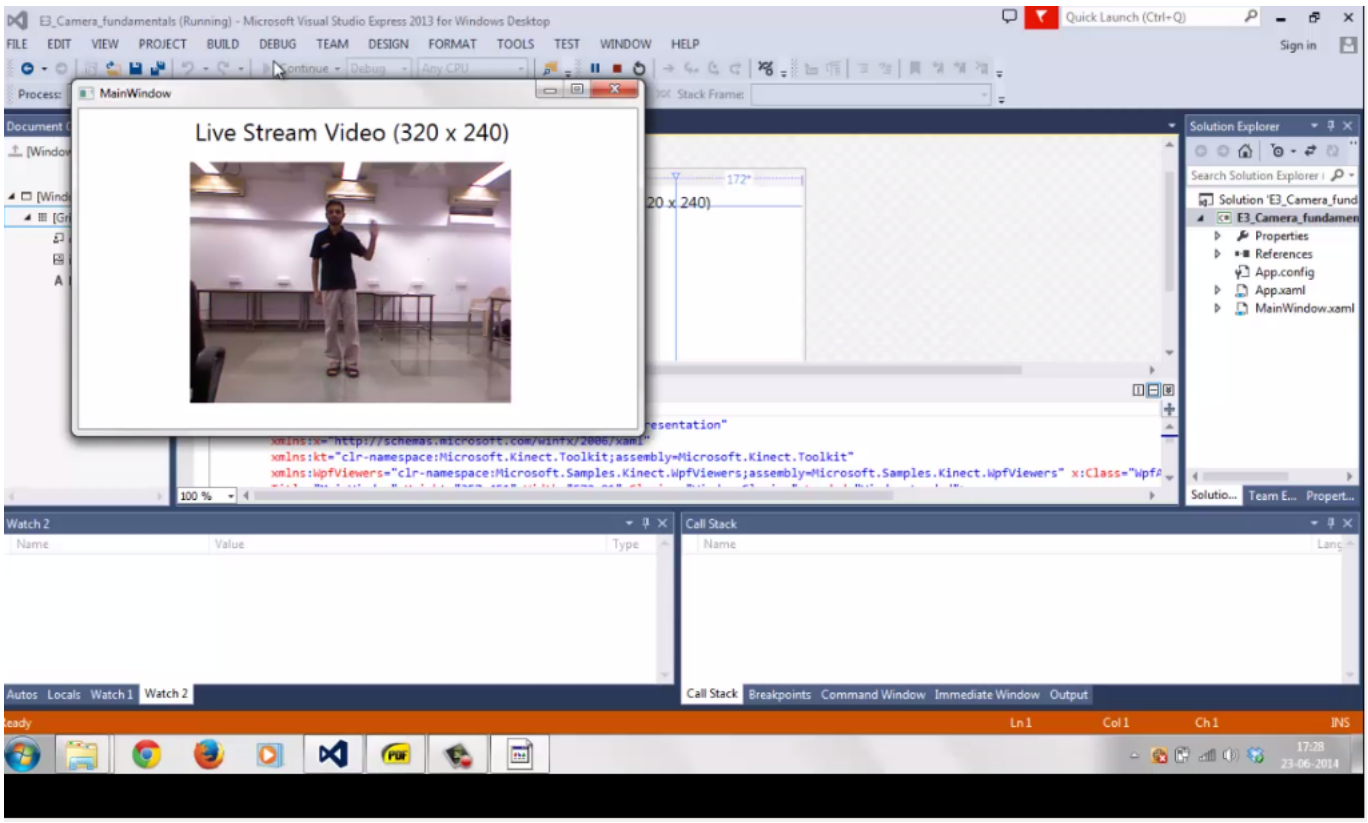
The experiment is available in the folder **E3_Camera_Fundamentals** provided.

4.3.4 Instructions

1. Click on the 'start' button to run the program.
2. You will see the live camera stream from the kinect camera in 320x240 resolution.
3. When done experimenting either close the main window or click on the 'Halt' button to stop the execution of the code.

4.3.5 Experiment Outputs





4.3.6 Conclusion

- The experiment was successfully conducted to obtain the live camera stream captured by the kinect camera and to display it on screen.

4.4 Skeleton Tracking Fundamentals

4.4.1 Aim of the experiment

The aim of this experiment is to obtain the joint information of 20 joints of the body in the frame captured by the kinect and display them on screen as a skeleton.

4.4.2 Components required

- Kinect sensor

4.4.3 Experiment description

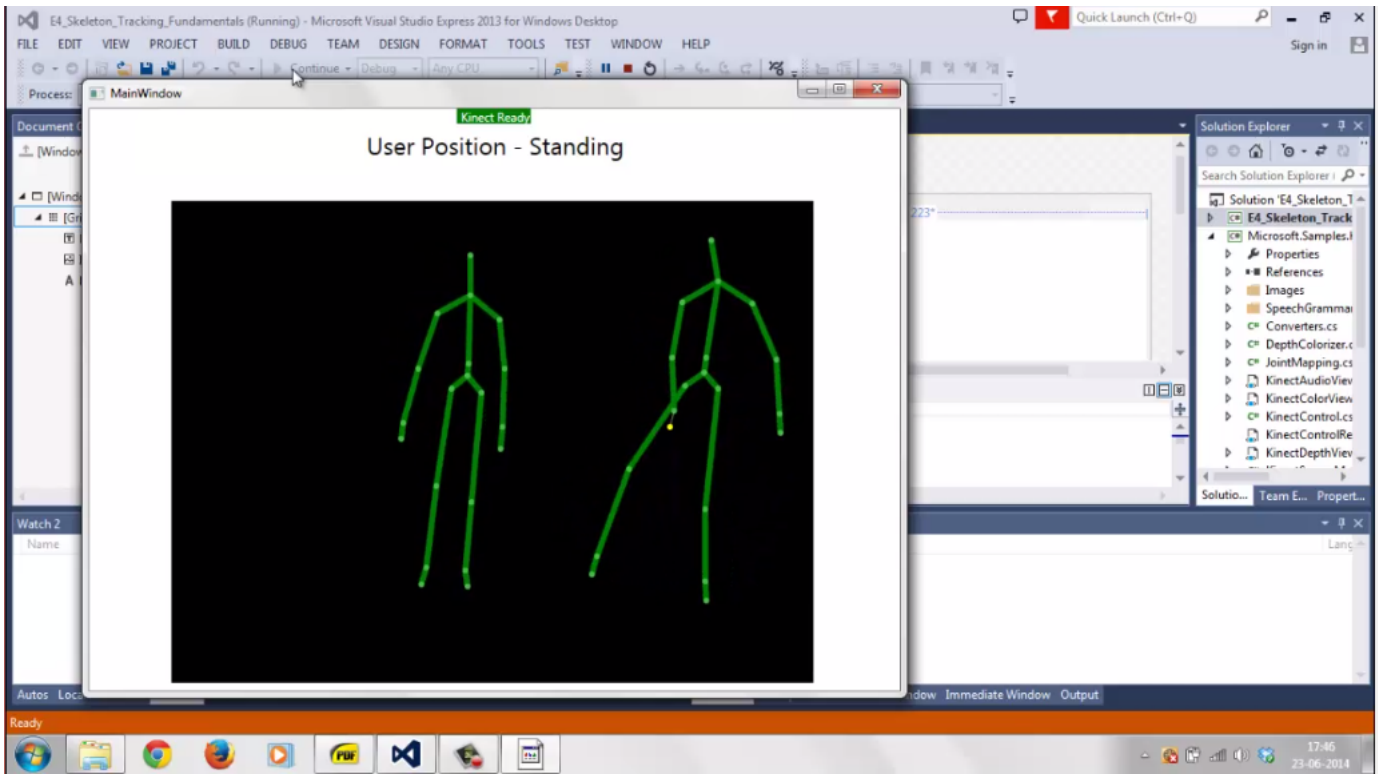
This experiment obtains the joint data of 20 joints corresponding to the user in the view of the kinect camera and display them on screen in form of a skeleton. The green dots represent the joints. The joints are connected by light green lines. If the skeleton goes out of the frame, it is indicated by a red brush in that direction.

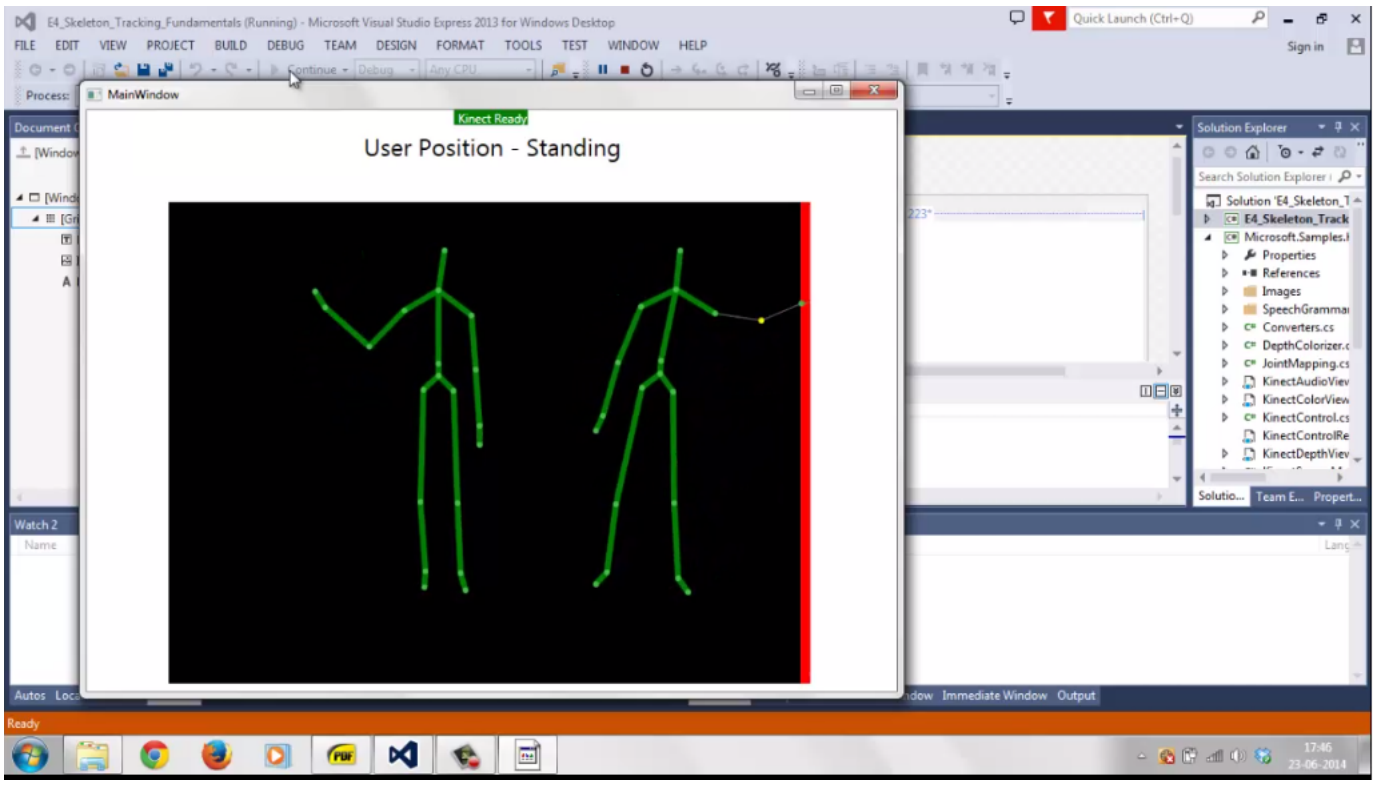
The experiment is available in the folder **E4_Skeleton_Tracking_Fundamentals** provided.

4.4.4 Instructions

1. Click on the 'start' button to run the program.
2. The user must stand at a distance such that the full skeleton is visible on screen.
3. You will see a skeleton corresponding to the user on the screen. The green dots represent the joints. The joints are connected by light green lines. If the skeleton goes out of the frame, it is indicated by a red brush in that direction.
4. When done experimenting either close the main window or click on the 'Halt' button to stop the execution of the code.

4.4.5 Experiment Outputs





4.4.6 Conclusion

- The experiment was successfully conducted to obtain the joint data of the user in the frame captured by the kinect and display a skeleton corresponding to it on screen.

4.5 Skeleton Tracking Firebird Left Right

4.5.1 Aim of the experiment

The aim of this experiment is to make the firebird move based on the position of the right elbow.

4.5.2 Components required

- Kinect sensor
- Firebird V Robot
- Zigbee Modules (2 nos.)
- Zigbee USB adapter

4.5.3 Experiment description

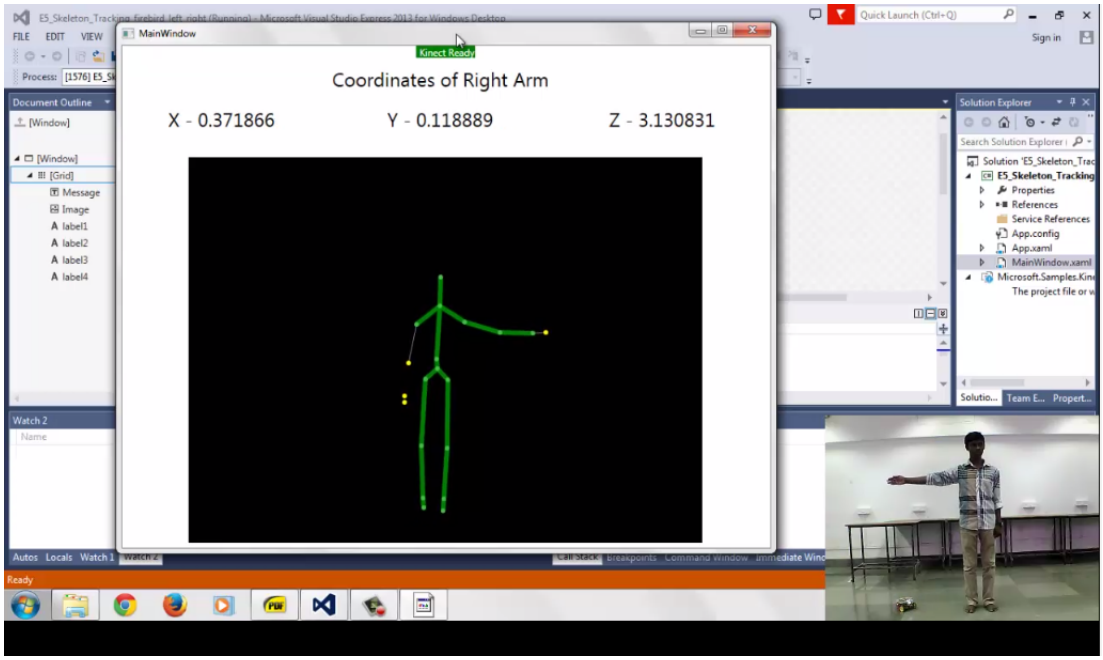
This experiment obtains the joint data of 20 joints corresponding to the user in the view of the kinect camera and display them on screen in form of a skeleton. The green dots represent the joints. The joints are connected by light green lines. If the skeleton goes out of the frame, it is indicated by a red brush in that direction. Then based on the coordinates of the right elbow, the firebird is sent a command via zigbee such that if the right elbow is in the right half of the image, the firebird moves right("56" is sent, '5' for stop, '6' for right), and if it is in the left half of the image, the firebird moves left("54" is sent, '5' for stop, '4' for left).

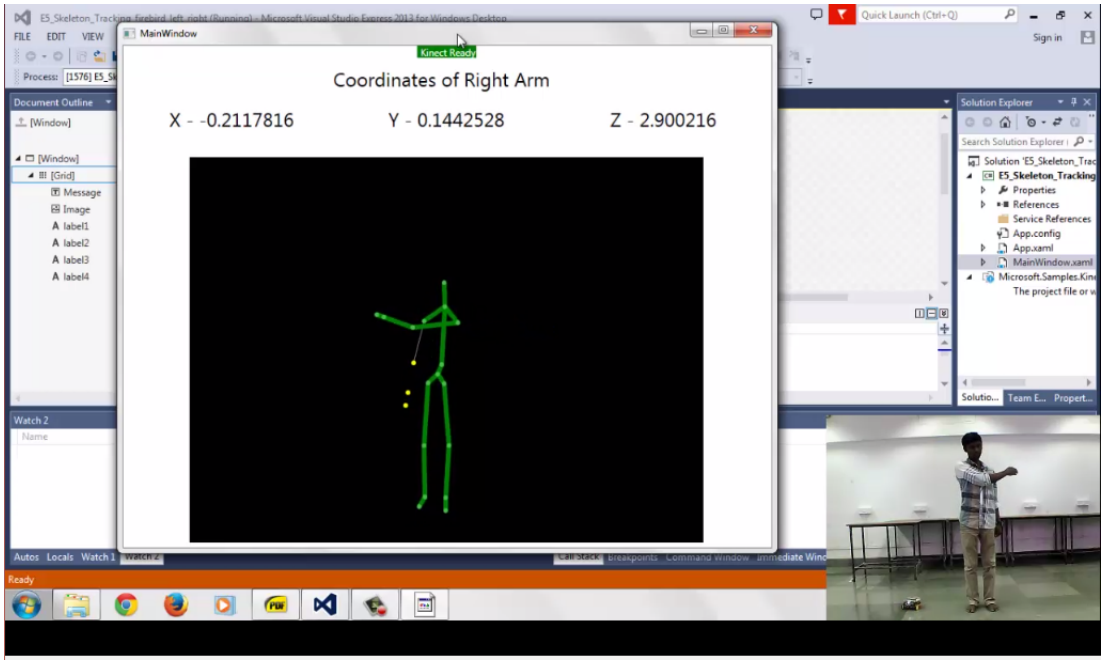
The experiment is available in the folder **E5_Skeleton_Tracking_Firebird_Left_Right** provided.

4.5.4 Instructions

1. Click on the 'start' button to run the program.
2. Turn on the firebird 5 robot loaded with the zigbee serial interfacing code and establish unicast connection between two zigbee modules as described in the manual.
3. The user must stand at a distance such that the full skeleton is visible on screen.
4. You will see a skeleton corresponding to the user on the screen. The green dots represent the joints. The joints are connected by light green lines. If the skeleton goes out of the frame, it is indicated by a red brush in that direction.
5. Move your right elbow to and fro between the right and left halves of the image and watch the firebird move right("56") and left("54").
6. When done experimenting either close the main window or click on the 'Halt' button to stop the execution of the code.

4.5.5 Experiment Outputs





4.5.6 Conclusion

- The experiment was successfully conducted to obtain the joint data of the user in the frame captured by the kinect and display a skeleton corresponding to it on screen.
- The right elbow joint coordinates were tracked in realtime and based on it, commands were sent to the firebird V.

4.6 Skeleton Tracking Firebird Front Back

4.6.1 Aim of the experiment

The aim of this experiment is to make the firebird move based on the change in position of the hip center joint.

4.6.2 Components required

- Kinect sensor
- Firebird V Robot
- Zigbee Modules (2 nos.)
- Zigbee USB adapter

4.6.3 Experiment description

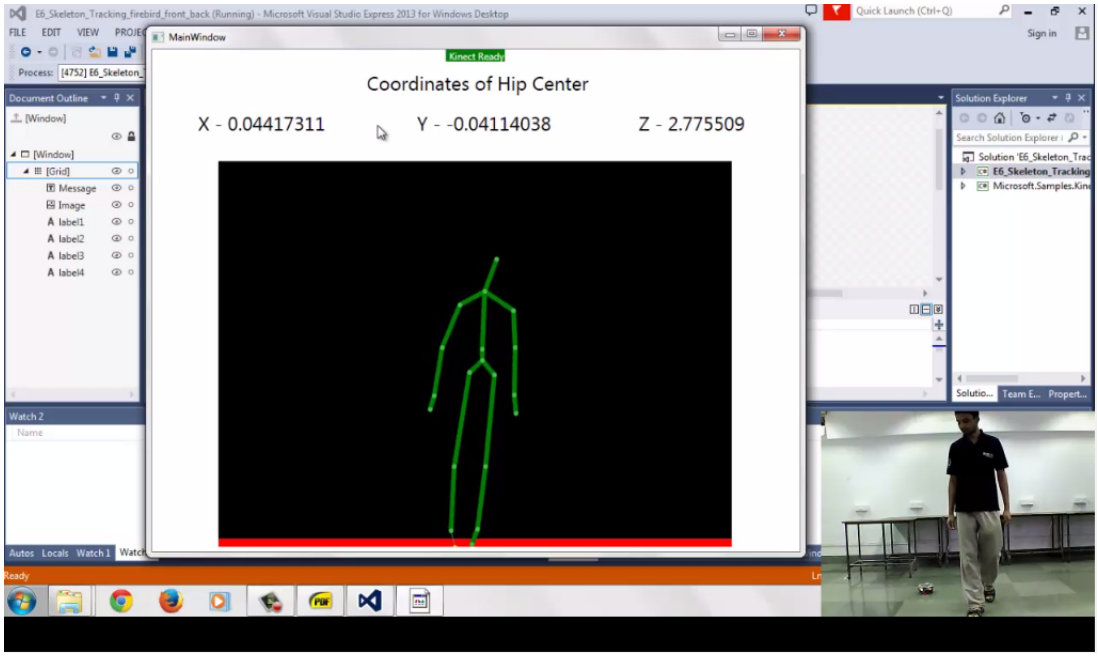
This experiment obtains the joint data of 20 joints corresponding to the user in the view of the kinect camera and display them on screen in form of a skeleton. The green dots represent the joints. The joints are connected by light green lines. If the skeleton goes out of the frame, it is indicated by a red brush in that direction. Then based on the motion of the hip center joint, the firebird is sent a command via zigbee such that if the hip center moves towards the kinect, the firebird moves forward("58" is sent, '5' for stop, '8' for forward), and if it moves away from the kinect, the firebird moves away("52" is sent, '5' for stop, '2' for reverse).

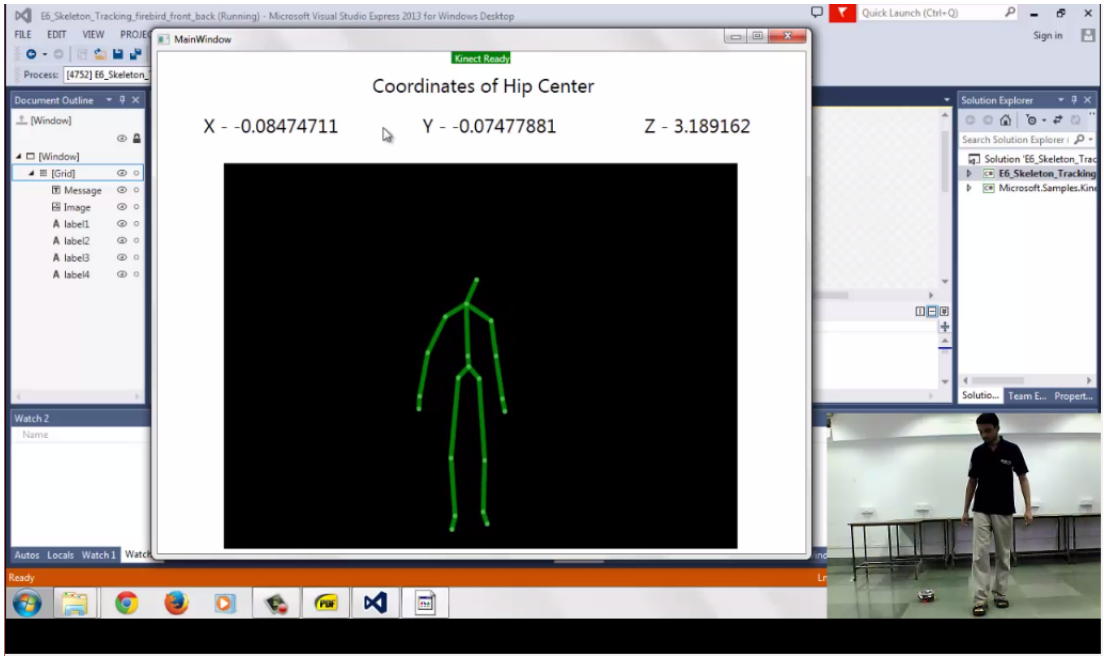
The experiment is available in the folder **E6_Skeleton_Tracking_Firebird_Front_Back** provided.

4.6.4 Instructions

1. Click on the 'start' button to run the program.
2. Turn on the firebird 5 robot loaded with the zigbee serial interfacing code and establish unicast connection between two zigbee modules as described in the manual.
3. The user must stand at a distance such that the full skeleton is visible on screen.
4. You will see a skeleton corresponding to the user on the screen. The green dots represent the joints. The joints are connected by light green lines. If the skeleton goes out of the frame, it is indicated by a red brush in that direction.
5. Move towards and away from the kinect and watch the firebird move forward("58") and reverse("52").
6. When done experimenting either close the main window or click on the 'Halt' button to stop the execution of the code.

4.6.5 Experiment Outputs





4.6.6 Conclusion

- The experiment was successfully conducted to obtain the joint data of the user in the frame captured by the kinect and display a skeleton corresponding to it on screen.
- The motion of the hip center joint was tracked in realtime and based on it, commands were sent to the firebird V.

4.7 Skeleton Tracking angle between joints

4.7.1 Aim of the experiment

The aim of this experiment is to obtain the angle between 3 body joints in the frame captured by the kinect.

4.7.2 Components required

- Kinect sensor.

4.7.3 Experiment description

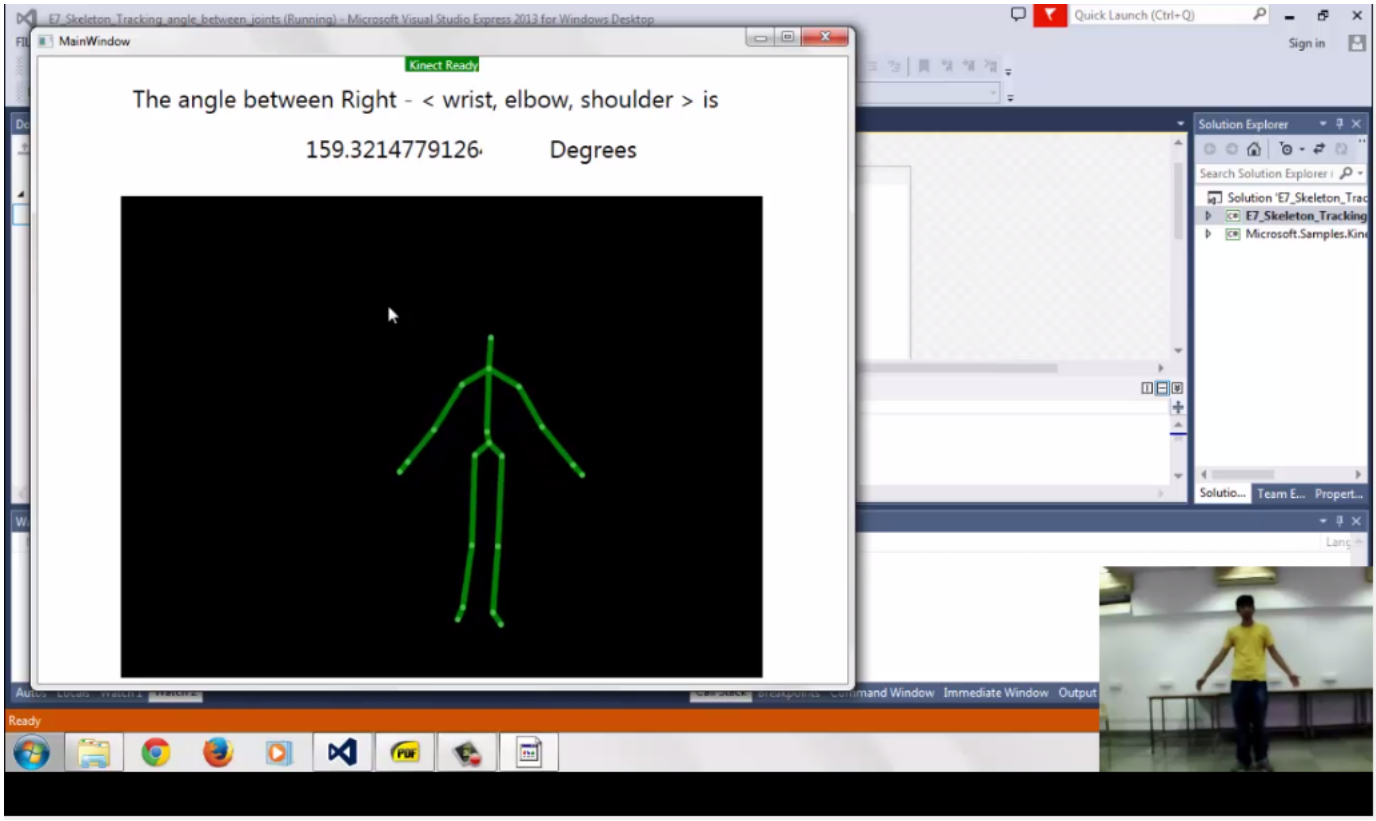
This experiment is used to determine the angle between 3 body joints and displayed on the screen. For the sake of simplicity the experiment is designed for right elbow, right wrist and right shoulder. However, the user can change this to any 3 joints of choice. To get the angle, Vector algebra in 3D is used for the purpose. The right elbow is used a fulcrum. The absolute x, y and z coordinates of the joints obtainted. Their relative distance is calculated. This gives us 2 vectors. Lets call them A and B. Then the angle between the 2 vectors is calculated using dot product concept: angle cos(x) = A.B / —A— x —B—

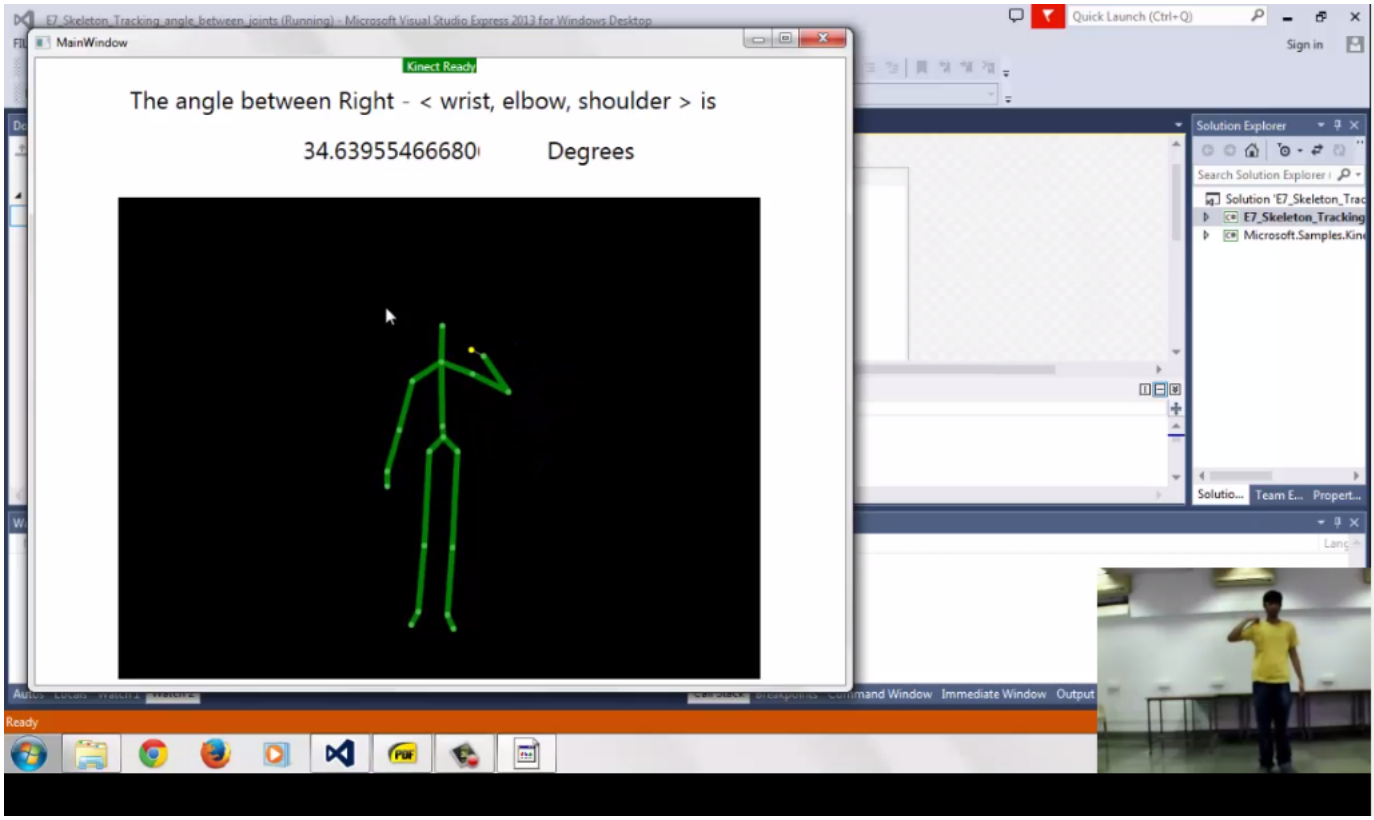
The experiment is available in the folder **E7_Skeleton_Tracking_angle_between_joints** provided.

4.7.4 Instructions

1. Click on the 'start' button to run the program.
2. The user must stand at a viable distance from the kinect so that the entire body is captured in the frame. In case any part of the skeleton is out of frame the indication is given by a red brush on that side.
3. The user can change the 3 joints to any sensible combination.
4. The experiment tracks the nearest skeleton in the frame hence in case of multiple users the response will be given to the nearest one to the kinect.
5. When done experimenting either close the main window or click on the 'stop' button to stop the execution of the code.

4.7.5 Experiment Outputs





4.7.6 Conclusion

The experiment was successfully conducted to obtain the angle between 3 body joints - rightelbow, rightwrist and rightshoulder.

4.8 Skeleton Tracking Hoverbutton Firebird

4.8.1 Aim of the experiment

The aim of this experiment is to make the firebird move based on the selected button on screen using the hand as a cursor.

4.8.2 Components required

- Kinect sensor
- Firebird V Robot
- Zigbee Modules (2 nos.)
- Zigbee USB adapter

4.8.3 Experiment description

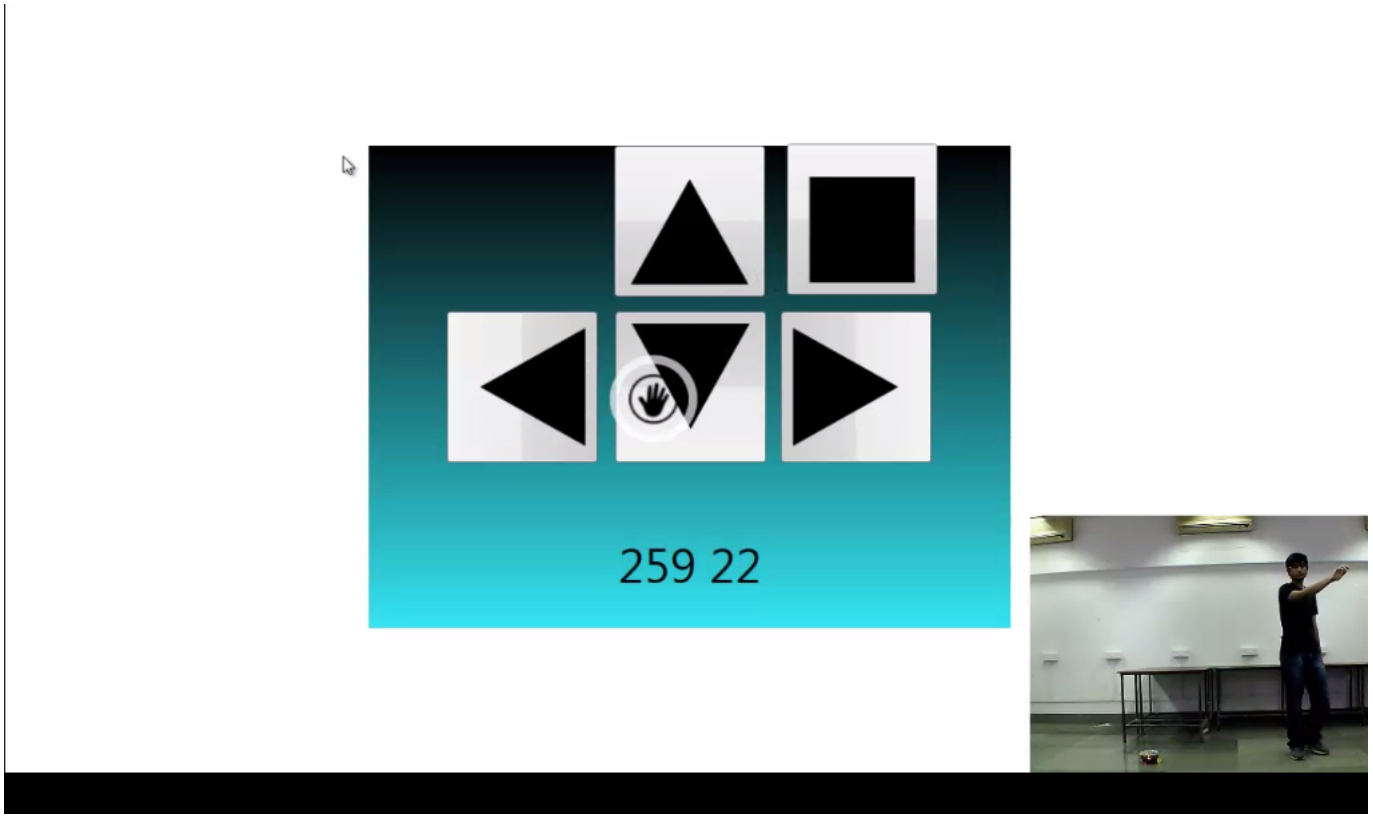
This experiment has an interface in which the pointer can be controlled by motion of the hand. Then based on the position of the cursor, the firebird is sent a command via zigbee such that if the right arrow button is selected, the firebird moves right("56" is sent, '5' for stop, '6' for right), if left arrow button is selected, the firebird moves left("54" is sent, '5' for stop, '4' for left), if the up arrow button is selected, the firebird moves forward("58" is sent, '5' for stop, '8' for forward), and if the down arrow button is selected, the firebird moves in reverse("52" is sent, '5' for stop, '2' for reverse). If the stop button is selected, the firebird stops("5" is sent to stop the firebird).

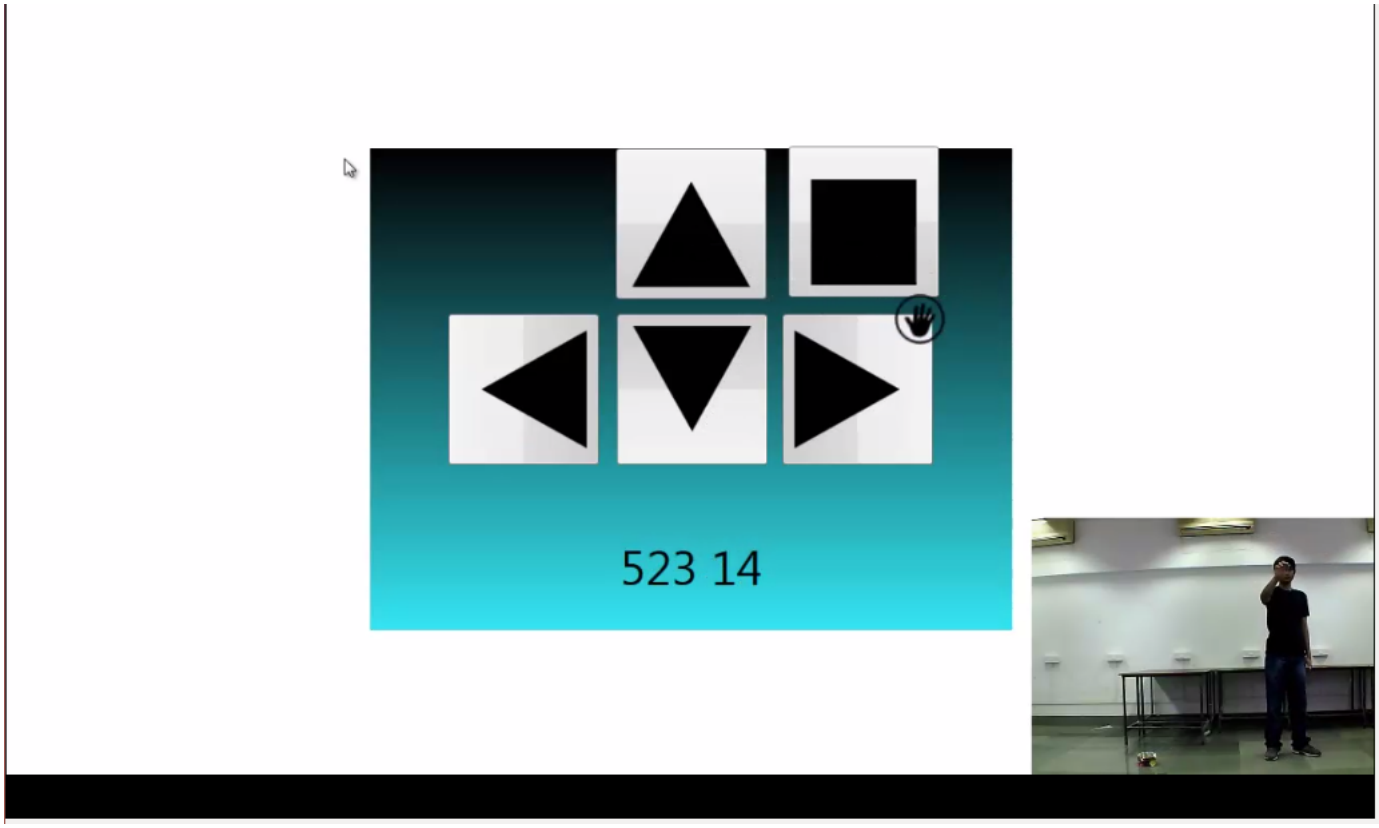
The experiment is available in the folder **E8_Skeleton_Tracking_Hoverbutton_Firebird** provided.

4.8.4 Instructions

1. Click on the 'start' button to run the program.
2. Turn on the firebird 5 robot loaded with the zigbee serial interfacing code and establish unicast connection between two zigbee modules as described in the manual.
3. The user must stand at a distance such that the full skeleton is in the frame of the kinect.
4. You will see a pointer on screen which moves in accordance with the movement of the right arm.
5. Select various buttons by hovering on them, right arrow button makes the firebird move right("56" is sent), left arrow button makes the firebird move left("54" is sent), up arrow button makes the firebird move forward("58" is sent), down arrow button makes the firebird move down("52" is sent), stop button makes the firebird stop("5" is sent).
6. When done experimenting either close the main window or click on the 'Halt' button to stop the execution of the code.

4.8.5 Experiment Outputs





4.8.6 Conclusion

- The experiment was successfully conducted to operate the pointer based on hand motion and interacting with the GUI with the pointer, then sending various commands to the firebird based on it.

4.9 Skeleton Tracking Complete

4.9.1 Aim of the experiment

The aim of this experiment is to demonstrate a simple application in which firebird follows human gesture

4.9.2 Components required

- Kinect sensor.
- Firebird V robot.
- Zigbee Modules (2 nos.)
- Zigbee USB adapter.

4.9.3 Experiment description

This experiment is used for simple gesture recognition by tracking the angles between the various body joints. The gestures are as follows :

- stand still - stop gesture
- raise right hand - turn right gesture
- raise left hand - turn left gesture
- lean forward - move forward gesture
- raise both hands - move back gesture

The commands transmitted serially to the firebird are as follows :

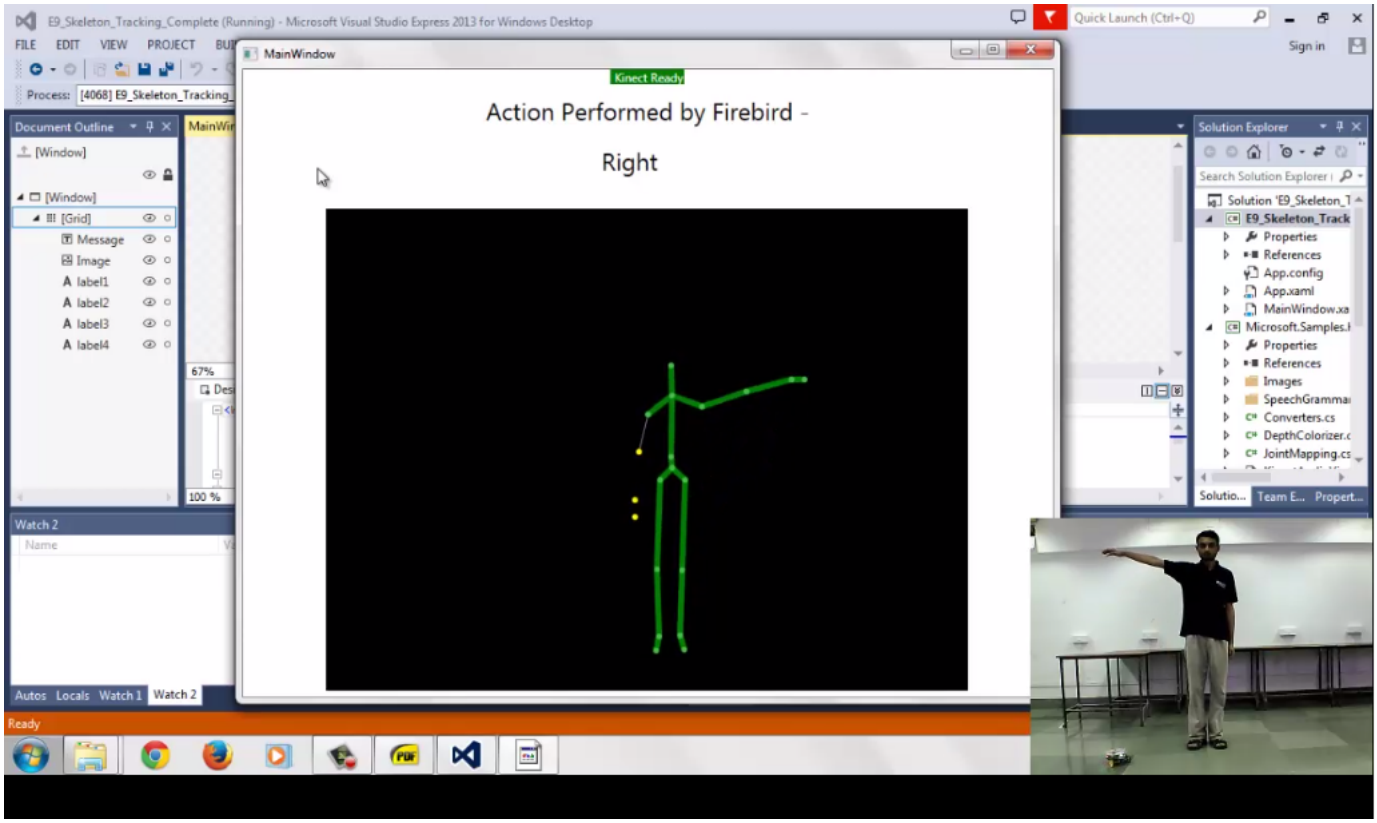
- 2 - backward
- 4 - left
- 5 - stop
- 6 - right
- 8 - forward
- 9 - buzzer on

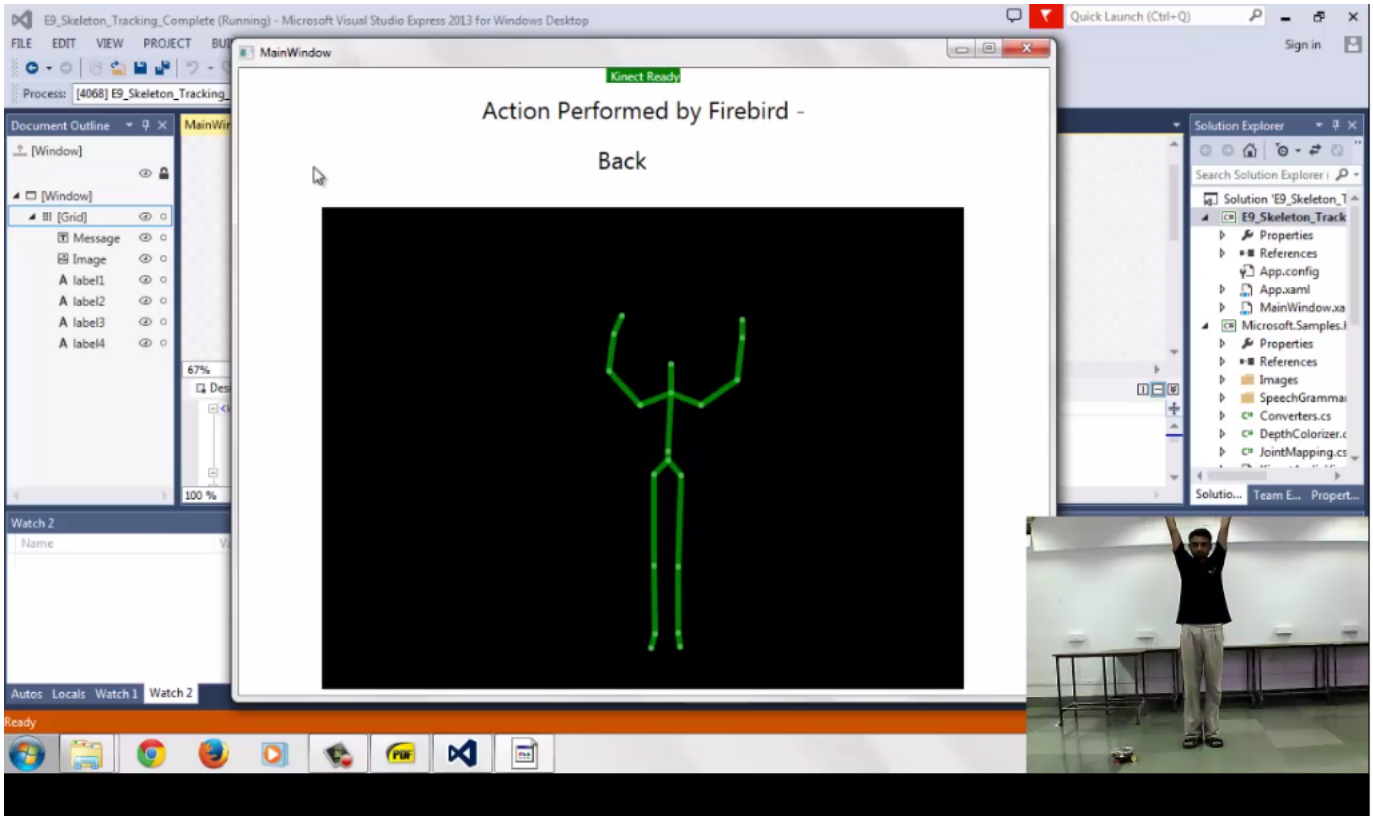
The experiment is available in the folder **E9_Skeleton_Tracking_Complete** provided.

4.9.4 Instructions

1. Click on the 'start' button to run the program.
2. Turn on the Firebird V robot loaded with **Interfacing_Kinect_To_Firebird_Using_Zigbee** program provided.
3. The user must stand at a viable distance from the kinect so that the entire body is captured in the frame. In case any part of the skeleton is out of frame the indication is given by a red brush on that side.
4. The user can change the 3 joints to any sensible combination.
5. The experiment tracks the nearest skeleton in the frame hence in case of multiple users the response will be given to the nearest one to the kinect.
6. Perform the gestures as mentioned in the theory above and observe the response on the Firebird V robot.
7. When done experimenting either close the main window or click on the 'stop' button to stop the execution of the code.

4.9.5 Experiment Outputs





4.9.6 Conclusion

The experiment was successfully completed and the response of Firebird V robot was observed for the various gestures performed.

4.10 Speech Recognition

4.10.1 Aim of the experiment

The aim of this experiment is to make the Firebird V robot responsive to the speech commands of the user.

4.10.2 Components required

- Kinect sensor.
- Firebird V robot.
- Zigbee Modules (2 nos.)
- Zigbee USB adapter.

4.10.3 Experiment description

This experiment is designed for voice recognition and deciphering the commands given by the user. This is followed by interfacing to firbird V to follow the commands. The commands include speaking :

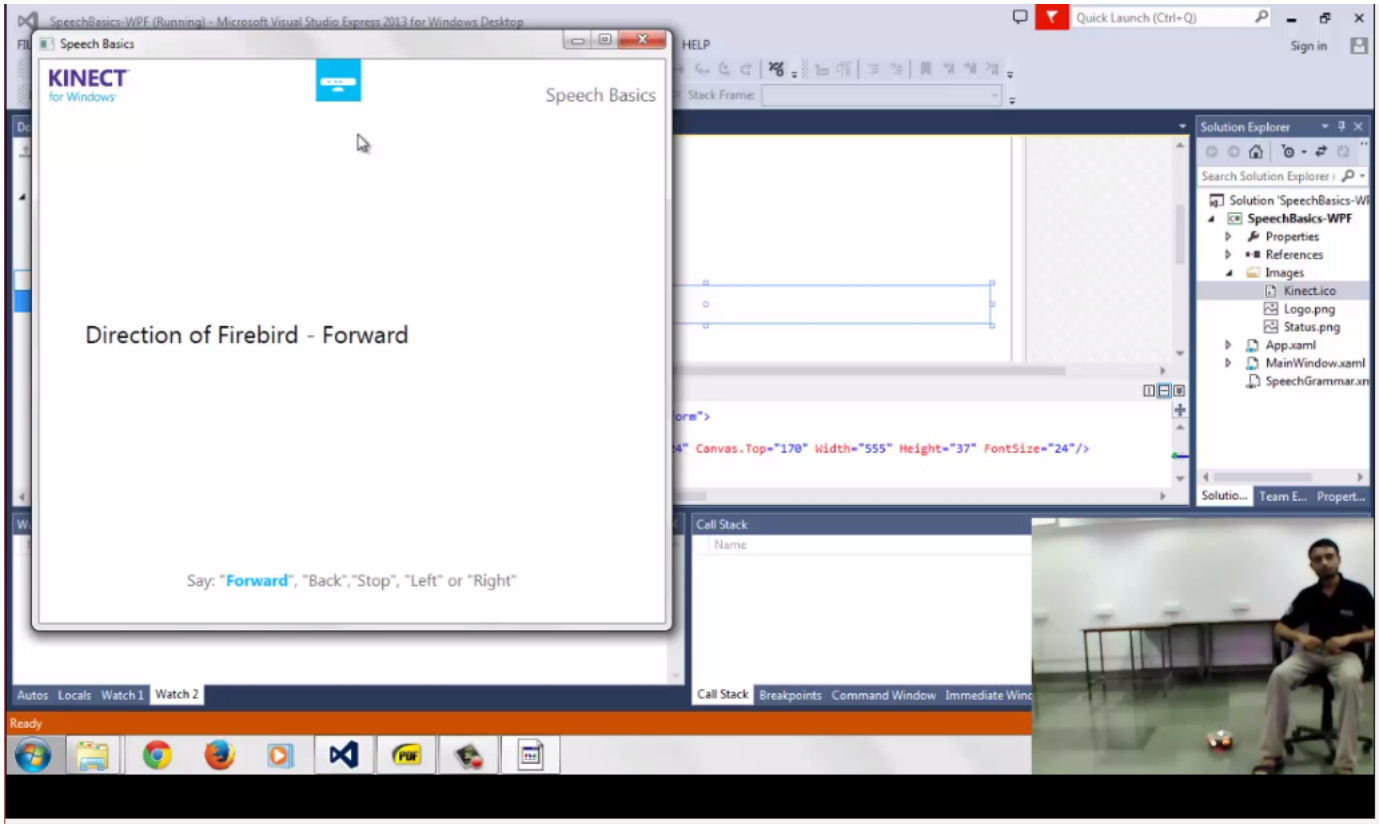
- "Forward" - To move the robot forward
- "Back" - To move the robot backward
- "Left" - To move the robot left
- "Right" - To move move the robot right
- 2 - backward
- 4 - left
- 5 - stop
- 6 - right
- 8 - forward
- 9 - buzzer on

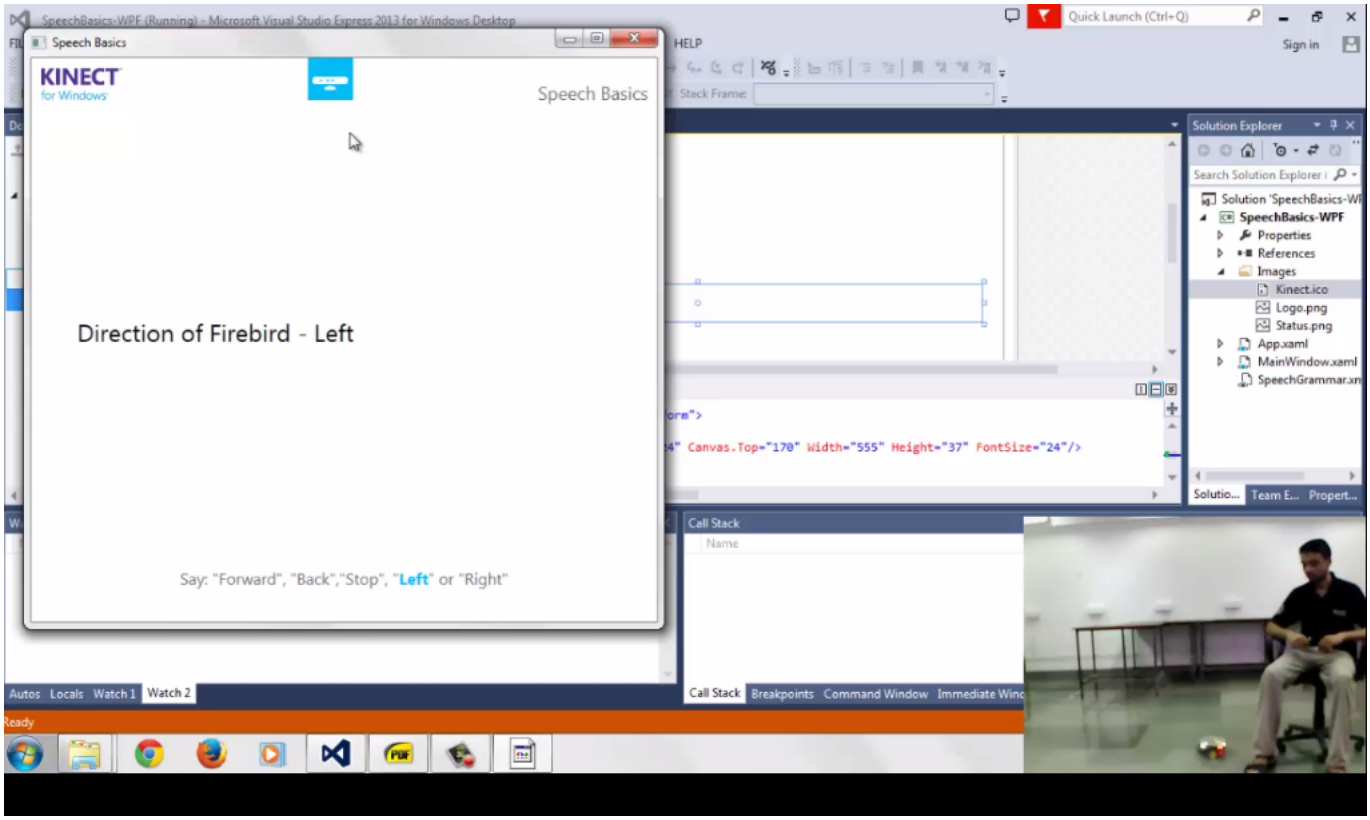
The experiment is available in the folder **E10_Voice_Recognition_firebird** provided.

4.10.4 Instructions

1. Click on the 'start' button to run the program.
2. Turn on the Firebird V robot loaded with **Interfacing_Kinect_To_Firebird_Using_Zigbee** program provided.
3. The user can modify the existing grammer available in the **SpeechGrammer.xml file**.
4. Speak the commands mention in the Theory and observe the response on the Firebird V.
5. The commands need to be spoken with confidence (threshold = 0.25) and must be pronounced correctly.
6. The experiment works with multiple users. However, multiple users giving simultaneous commands can lead to unpredictable results.
7. Background noise must be minimal or avoided if possible.
8. When done experimenting either close the main window or click on the 'stop' button to stop the execution of the code.

4.10.5 Experiment Outputs





4.10.6 Conclusion

The experiment was successfully completed to make the Firebird V robot responsive to simple voice commands given by the user.

4.11 Tilt Demo

4.11.1 Aim of the experiment

The aim of this experiment is to change the elevation angle of the kinect.

4.11.2 Components required

- Kinect sensor

4.11.3 Experiment description

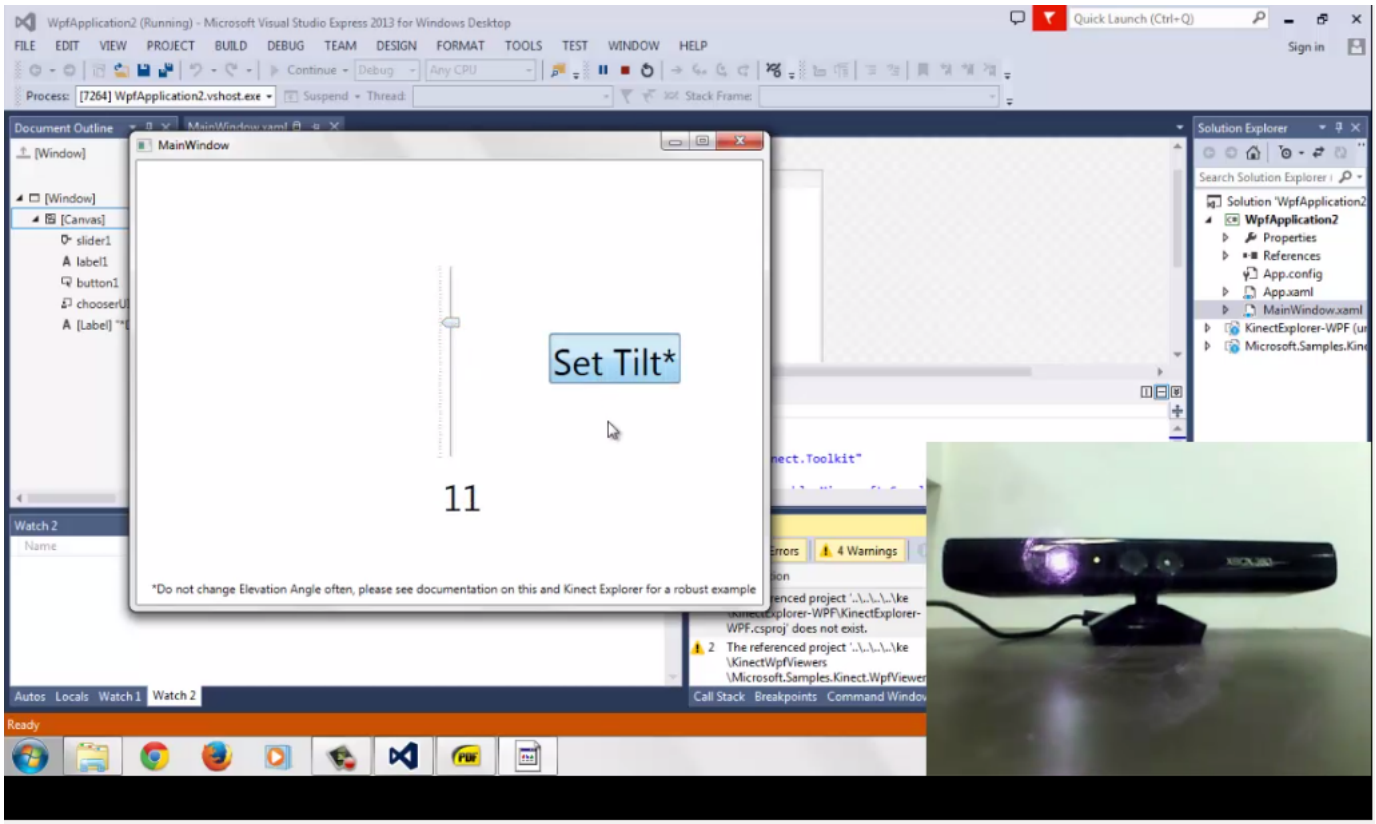
This experiment changes the elevation angle of the kinect in the range -27 degrees to +27 degrees based on user input.

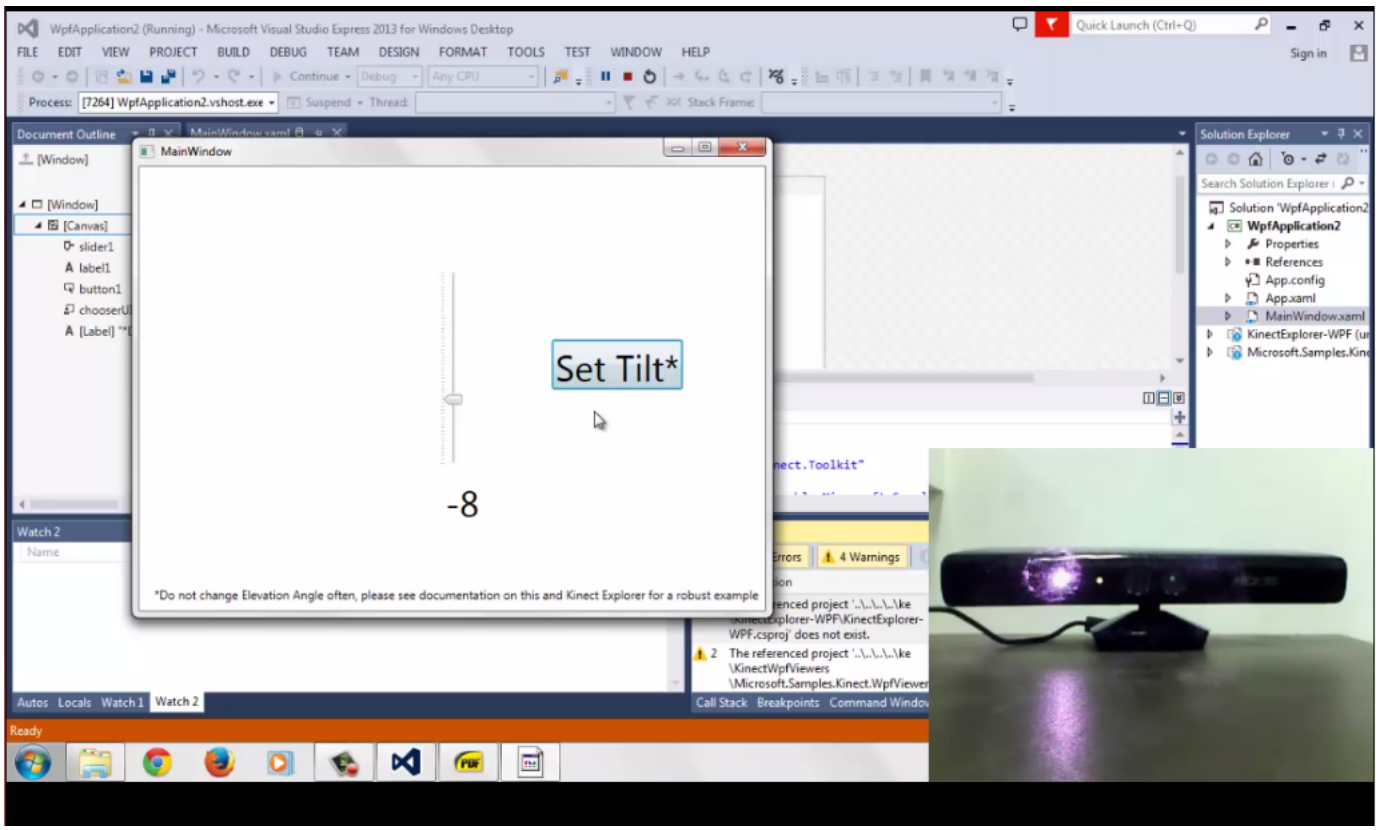
The experiment is available in the folder **E11_Tilt_Demo** provided.

4.11.4 Instructions

1. Click on the 'start' button to run the program.
2. Set the desired elevation angle on the slider and click the "set tilt" button to set the kinect's elevation angle to that value.
3. When done experimenting either close the main window or click on the 'Halt' button to stop the execution of the code.

4.11.5 Experiment Outputs





4.11.6 Conclusion

- The experiment was successfully conducted to change the kinect's elevation angle to a desired value.

Note : The next section deals with Experiments designed on linux. Before starting the experiments navigate to **examples** folder in the **libfreenect** directory and copy the code files of the 3 experiments (E12, E13 and E14) in the same. Open the **CMakeLists.txt** file and make the following changes in the respective sections :

- file(GLOB SRC_STANDARD chunkview.c glview.c hiview.c regview.c E12_Depth_Tracking_on_screen.c E13_Depth_Tracking_on_Firebird.c E14_Tilt_demo.c)
- add_executable(freenect-E12_Depth_Tracking_on_screen E12_Depth_Tracking_on_screen.c)
- add_executable(freenect-E13_Depth_Tracking_0n_Firebird E13_Depth_Tracking_on_Firebird.c)
- add_executable(freenect-E14_Tilt_demo E14_Tilt_demo.c)
- target_link_libraries(freenect-E13_Depth_Tracking_on_Firebird freenect)
- target_link_libraries(freenect-E12_Depth_Tracking_on_screen freenect)
- target_link_libraries(freenect-E14_Tilt_demo freenect freenect_sync)
- target_link_libraries(freenect-E12_Depth_Tracking_on_screen freenect $OPENGL_LIBRARIES $GLUT_LIBRARY $CMAKE_THREAD_LIBS_INIT $MATH_LIB)
- target_link_libraries(freenect-E13_Depth_Tracking_on_Firebird freenect $OPENGL_LIBRARIES $GLUT_LIBRARY $CMAKE_THREAD_LIBS_INIT $MATH_LIB)
- target_link_libraries(freenect-E14_Tilt_demo freenect_sync $CMAKE_THREAD_LIBS_INIT $MATH_LIB)
- install (TARGETS freenect-glview freenect-regview freenect-hiview freenect-chunkview freenect-E12_Depth_Tracking_on_screen freenect-E13_Depth_Tracking_on_Firebird DESTINATION bin)
- install (TARGETS freenect-glpclview freenect-E14_Tilt_demo DESTINATION bin)

4.12 Depth tracking on screen

4.12.1 Aim of the experiment

The aim of this experiment is to obtain the depth information of every pixel in the frame captured by the kinect.

4.12.2 Components required

- Kinect sensor.

4.12.3 Experiment description

This experiment obtains the depth value of every pixel in the frame. Based on this obtained information a color coding scheme is designed. If the pixel is at a distance of less than 900mm (and greater than 800mm) it is assigned the color Red. If the pixel is at a distance greater than 900mm and less than 200mm the color assigned to it is Green. If the pixel is at a distance greater than 2000mm then it is assigned the color Blue. Thus the output frame comprises of Blue Green or Red colored pixels depending upon its distance from the kinect. The experiment displays the live stream video along side the depth output window for simplicity.

The experiment is available in the folder **E12_Depth_Tracking_on_screen_linux** provided.

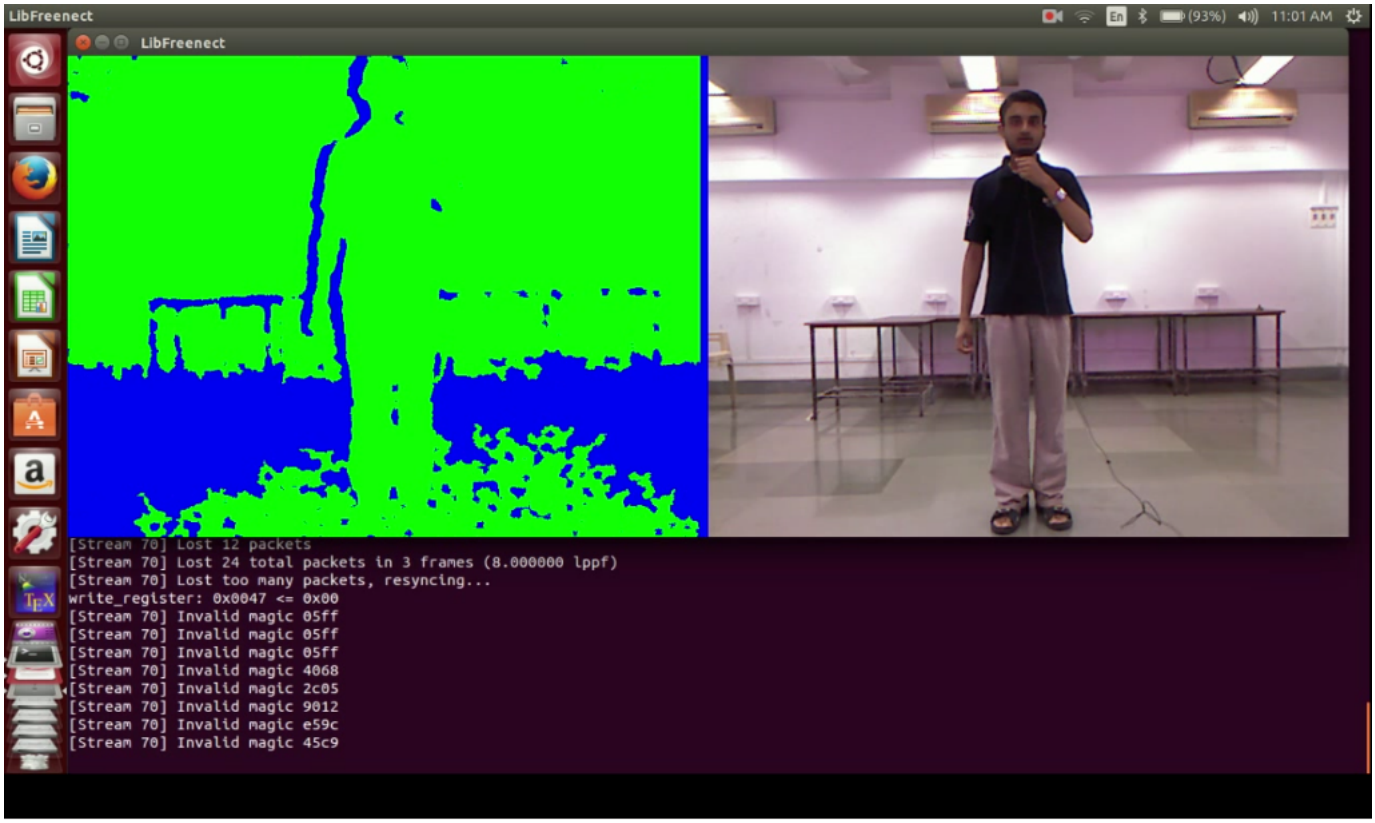
4.12.4 Instructions

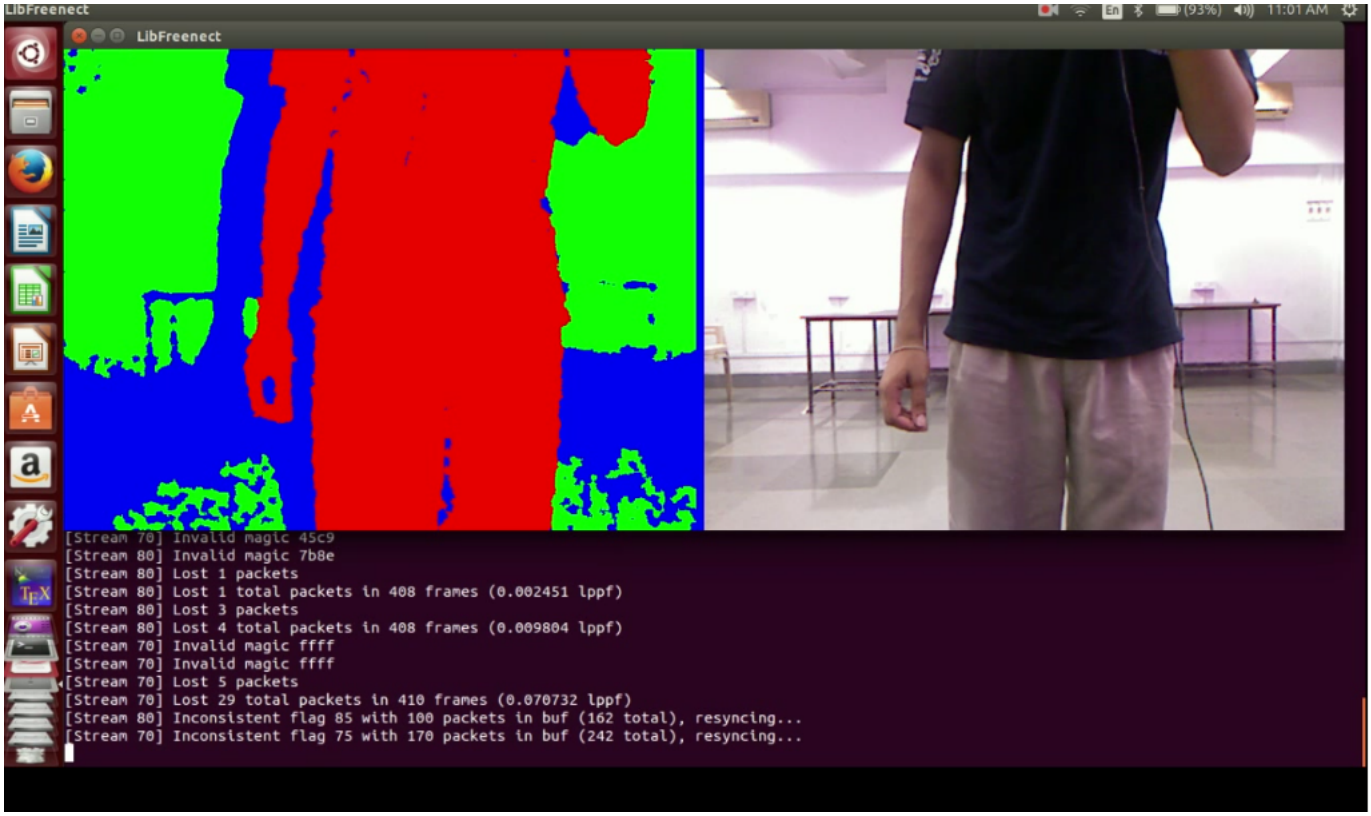
1. Open a new terminal and navigate to the **build** folder in the **libfreenect**.
2. type the following sequence of commands :

cmake ..
make
sudo make install
sudo freenect-E12_Depth_Tracking_on_screen_linux

3. Make sure that the background doesnot contain windows because sunlight (or any source of light) is interpreted being within the 900mm range. The violation of this step, however, will not affect the functionality of this experiment.
4. The user must stand at a distance of atleast 800mm from the kinect. This is the lower threshold of the kinect itself, to get depth data.

4.12.5 Experiment Outputs





4.12.6 Conclusion

The experiment was successfully conducted to obtain the depth information of every pixel in the frame captured by the kinect.

4.13 Depth tracking with firebird

4.13.1 Aim of the experiment

The aim of this experiment is to obtain the depth information of every pixel in the frame captured by the kinect and to make Firebird V responsive to the variation of the depth of the center pixel in the frame.

4.13.2 Components required

- Kinect sensor.
- Firebird V robot.
- Zigbee Modules (2 nos.)
- Zigbee USB adapter.

4.13.3 Experiment description

This experiment obtains the depth value of every pixel in the frame. Based on this obtained information a color coding scheme is designed. If the pixel is at a distance of less than 900mm (and greater than 800mm) it is assigned the color Red. If the pixel is at a distance greater than 900mm and less than 200mm the color assigned to it is Green. If the pixel is at a distance greater than 2000mm then it is assigned the color Blue. Thus the output frame comprises of Blue Green or Red colored pixels depending upon its distance from the kinect. Based on the distance of the center pixel from the kinect a command is sent to the Firebird V robot to move forward or backward based on the movement of the center pixel in the frame. The commands transmitted serially to the firebird are as follows :

- 2 - backward
- 8 - forward
- 5 - Stop

The experiment is available in the folder **E13_Depth_Tracking_on_firebird_linux** provided.

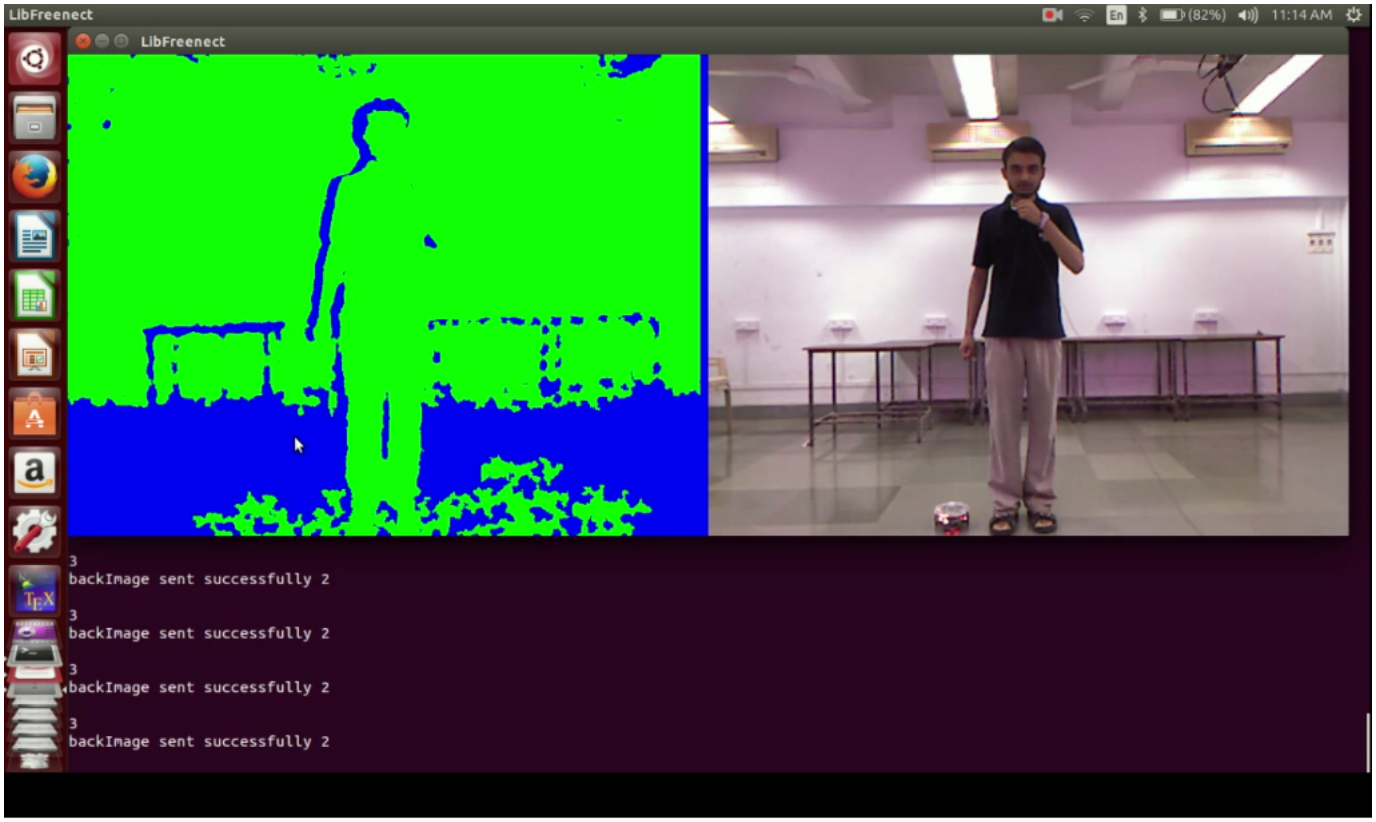
4.13.4 Instructions

1. Open a new terminal and navigate to the **build** folder in the **libfreenect**.
2. type the following sequence of commands :

cmake ..
make
sudo make install
sudo freenect-E13_Depth_Tracking_on_firebird_linux

3. Make sure that the background doesnot contain windows because sunlight (or any source of light) is interpreted being within the 900mm range. The violation of this step, however, will not affect the functionality of this experiment.
4. The user must stand at a distance of atleast 800mm from the kinect. This is the lower threshold of the kinect itself, to get depth data.
5. Observe the response of Firebird V robot with the variation of distance of teh center pixel in the frame.

4.13.5 Experiment Outputs



4.13.6 Conclusion

The experiment was successfully conducted to obtain the depth information of every pixel in the frame captured by the kinect. The response of the Firebird V robot with the movement of the center pixel in the frame was observed.

4.14 Tilt Demo

4.14.1 Aim of the experiment

The aim of this experiment is to obtain the depth information of every pixel in the frame captured by the kinect and to make Firebird V responsive to the variation of the depth of the center pixel in the frame.

4.14.2 Components required

- Kinect sensor.

4.14.3 Experiment description

This experiment takes the angle of tilt as an input from the user and tilts the KInect sensor by that much angle (perpendicular to the ground). The tilt angle can be anywhere between -27 degrees to 27 degrees. Any invalid input outside this range is immediately prompted to the user.

The experiment is available in the folder **E14_Tilt_Demo_linux** provided.

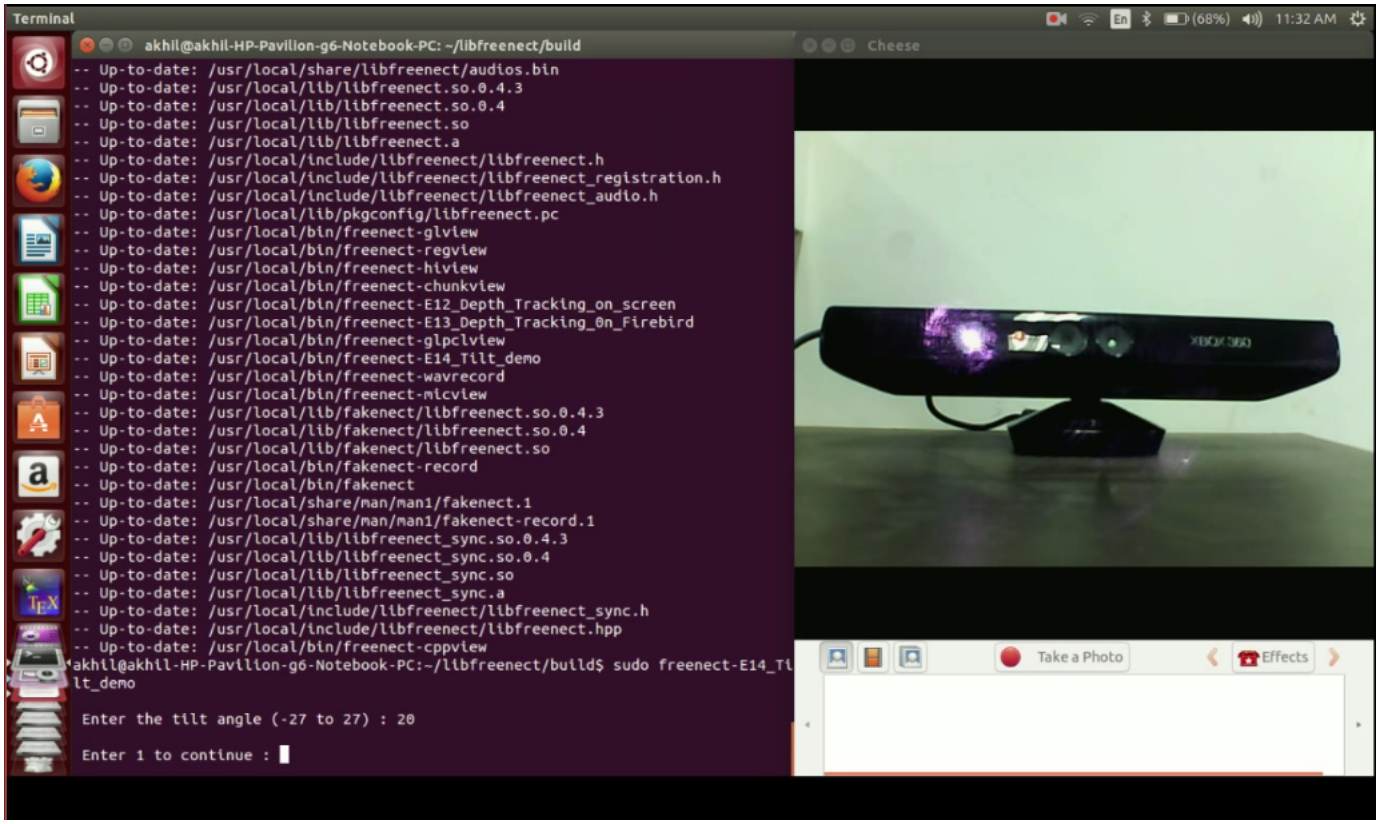
4.14.4 Instructions

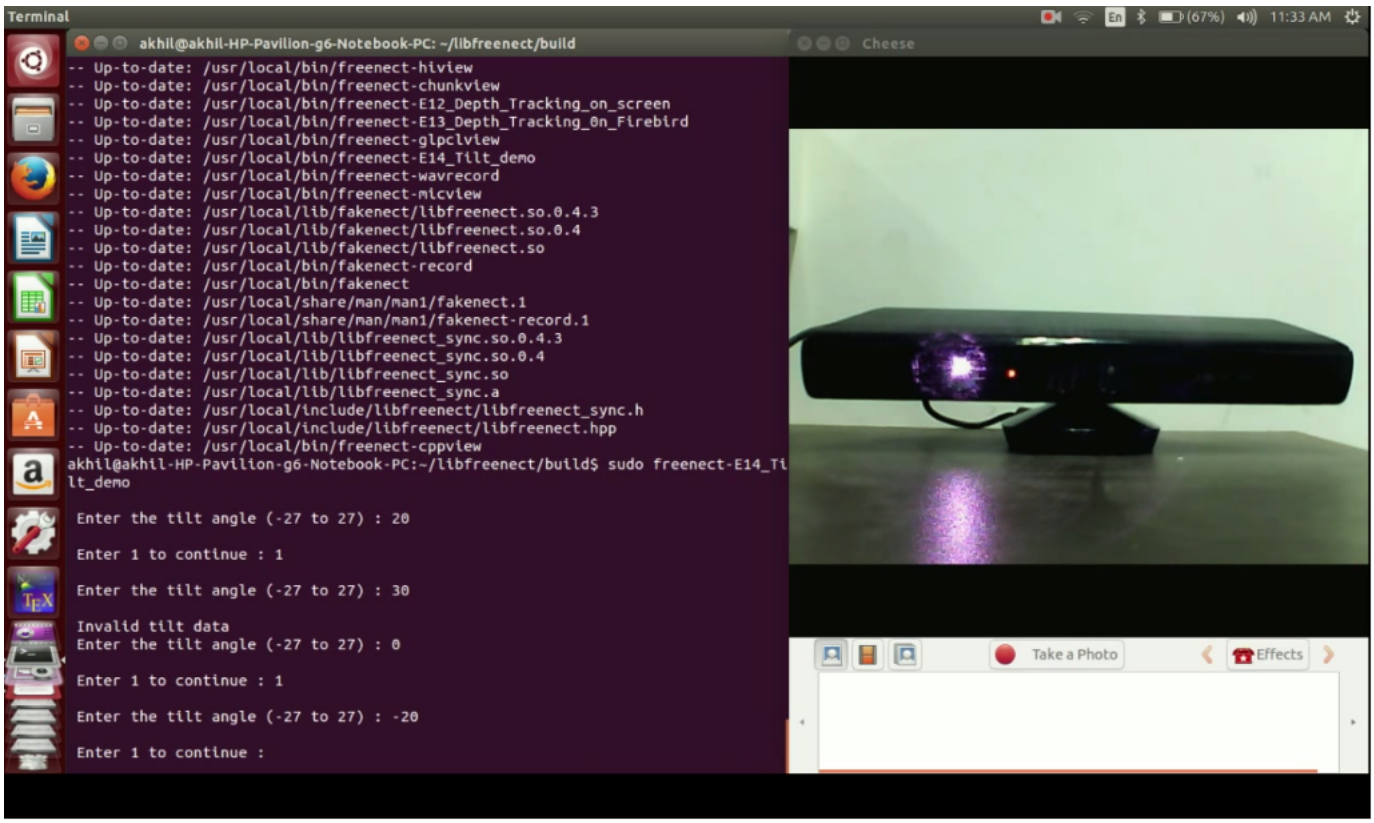
1. Open a new terminal and navigate to the **build** folder in the **libfreenect**.
2. type the following sequence of commands :

cmake ..
make
sudo make install
sudo freenect-E14_Tilt_demo_linux

3. Enter a value between -27 to 27 degrees and observe the tilting of the kinect.

4.14.5 Experiment Outputs





4.14.6 Conclusion

The experiment was successfully conducted to tilt the kinect in the specified angle within the valid range.

Bibliography

[1] http://www.microsoft.com/en-in/download/details.aspx?id=40278

[2] http://openkinect.org/wiki/Getting_Started

[3] http://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx

[4] http://learning.codasign.com/index.php?title=Skeleton_Tracking_with_the_Kinect

[5] https://www.youtube.com/watch?v=AwIlr98YZgk#t=31&hd=1

[6] http://members.ee.net/brey/Serial.pdf

[7] http://www.dotnetfunda.com/articles/show/2069/how-to-track-skeleton-joins-using-kinect

[8] http://channel9.msdn.com/coding4fun/kinect/Tracking-skeleton-joins-code-sample

[9] http://social.msdn.microsoft.com/Forums/en-US/3f73ac86-9793-4586-9eb3-3cf2aa55fc77/c-write-kinect-joint-positions-xyz-to-txt-file?forum=kinectsdk

[10] http://stackoverflow.com/questions/14204902/record-audio-with-kinect

[11] http://channel9.msdn.com/coding4fun/kinect/Introduction-to-Kinect-Speech-Recognition