1.INTRODUCTION

1.1. CONTEXT

There has always been considered a challenge, the development of a natural interaction interface, where people interact with technology as they are used to interact with the real world. A hand free interface, based only on human gestures, where no devices are attached to the user, will naturally immerse the user from the real world to the virtual environment.

Microsoft Kinect sensor device brings the long-expected technology to naturally interact with graphical interfaces to the masses. The user interacts having no physical device in his hands or attached to his body. Kinect captures the users' movements without the need of a controller, but through a Natural User Interface, using just gestures and spoken commands.

1.2 MOTIVATION

There is always a need to communicate using sign languages, such as chatting with speech and hearing challenged people. Additionally, there are situations when silent communication is preferred: for example, during an operation, a surgeon may gesture to the nurse for assistance. It is hard for most people who are not familiar with a sign language to communicate without an interpreter. Thus, software that transcribes symbols in sign languages into plain text can help with real time communication, and it also provides interactive training for people to learn a sign language. Gesture recognition has become an important research field with the current focus on interactive emotion recognition and HGR.

Traditionally, gesture recognition requires high quality Etereoscopic cameras and complicated computer vision algorithms to recogni2e hand signals; the systems often turn out to be expensive and require extensive setup. Microsoft Kinect provides an inexpensive and easy way for real-time user interaction. Kinect, originally designed for gaming on the JvTicrosoft Xbox platform, uses a color sensor and a depth sensor to capture color (RGB)

images and the associated depth (distance) data. It allows the development of algorithms that classify and perform recognition of the image data.

The software driver released by Microsoft called Kinect Software Development Kit (SDK) with Application Programming Interfaces (API) give access to raw sensor data streams as well as skeletal tracking. However, there is no hand specific data available for gesture recognition, although it does include information of the joints between hands and arms. Previous researches on computer vision and hand detection have established solid groundwork for gesture recognition. However, only a few Kinect based systems were developed for HGR and only a few gestures were recognized.

1.3 PROBLEM STATEMENT

To build an animatronic robotic arm, using the Arduino platform, whose motions are controlled by the operator via a Kinect device.

1.4 OBJECTIVE

The Objective of this project is to ease the job of a human operator by getting the work done by the robot simply by gesturing the actions that need to be done by the robot.

1.5 SCOPE

In this project we use the Microsoft Kinect sensor to communicate with the robotic arm. Communication between the human and the robot happens through the actions/gestures which the user wants the robot to perform. The robot imitates the users' actions which are sensed by the Kinect sensors. The values obtained from the gestures captured are sent to the robot via serialport communication. This way the user is capable of making the robot perform any action or job that he needs the robot to perform. In this manner, one can control the robotic arm to perform simple tasks like picking up a small lightweight object and dropping it off. Other scalable applications would be tightening a screw, defusal of a bomb, cleaning up of hazardous places etc.

2. LITERATURE SURVEY

2.1 What's New in version 1.7 of the SDK and the Developer Toolkit

http://msdn.microsoft.com/

Introducing new Kinect Interactions

Kinect SDK 1.7 is built with new Interactions framework which provides pre-packaged, reusable components that allow for even more exciting interaction possibilities. These components are supplied in both native and managed packages for maximum flexibility, and are also provided as a set of WPF controls. Among the new features are:

Press for Selection. This provides, along with the new KinectInteraction Controls, improved selection capability and faster interactions. If you're familiar with previous Kinect for Windows interaction capabilities, this replaces the hover select concept.

Grip and Move for Scrolling. This provides, along with the new KinectInteraction Controls, 1-to-1 manipulation for more precise scrolling, as well as large fast scrolls with a fling motion. If you're familiar with previous Kinect for Windows interaction capabilities, this replaces the hover scroll model.

New interactions work best with the following setup:

- User stands 1.5 - 2.0 meters away from the sensor
- Sensor mounted directly above or below the screen showing the application, and centered.
- Screen size < 46 inches
- Avoid extreme tilt angles
- Avoid lots of natural light and reflective materials for more reliable tracking

2.2 Application Development in Visual Studio

http://msdn.microsoft.com/en-us/library/

Visual Studio provides tools to design, develop, debug, and deploy applications.

The elements that are contained in the IDE.	Walkthrough: Explore the Visual Studio IDE with C# or Visual Basic
Creating solutions, projects, and files for your applications	Solution and Project Basics
Tracking work, managing bugs, using version control, and defining builds by using Team Foundation.	Using Team Foundation to Manage Development Processes
Using select designers and editors to create the UI for your applications	Designing Code and User Interfaces
Exploring code, using IntelliSense features, creating code snippets, and customizing your editing experience.	Writing Code in the Code and Text Editor
Creating, configuring, and managing builds.	Building Applications in Visual Studio
Setting breakpoints, handling exceptions, using edit and continue, and using other techniques for fine-tuning an application	Debugging in Visual Studio
Verifying code, analyzing performance, and using code metrics, among other optimizations	Improving Quality with Visual Studio Diagnostic Tools
Creating setup executables, packaging files, and publishing websites.	Deploying Applications and Components
Designing and developing applications to make it easier to distribute the application to a global audience	Globalizing and Localizing Applications

3. SYSTEM SPECIFICATION

3.1 SOFTWARE SPECIFICATIONS

- Operating System — Windows 7
- Programming Languages — C#, C++
- Applications Used — Visual Studio 2012, Kinect SDK
- Platform — Arduino(for Robotics)

3.2 HARDWARE REQUIREMENTS

- 32-bit (x86) or 64-bit (x64) processors
- Dual-core, 2.66-GHz or faster processor
- USB 2.0 bus dedicated to the Kinect
- 2 GB of RAM
- Graphics card that supports DirectX 9.0c

4. DETAILS

4.1 KINECT

Kinect is a motion sensing, input device by Microsoft for the Xbox 360 video game console and Windows PCs. Based around a webcam-style add-on peripheral for the Xbox 360 console, it enables users to control and interact with the Xbox 360 without the need to touch a game controller, through a natural user interface using gestures and spoken commands.

Microsoft released Kinect software development kit for Windows 7 on June 16, 2011. This SDK will allow developers to write Kinect apps in C++/CLI, C#, or Visual Basic .NET.

Previously known as “Project Natal”, Kinect was first announced on June 1, 2009, at E3. The name Natal means in Latin “to be born” and it was chosen because it reflects Microsoft's view of the project as "the birth of the next-generation of home entertainment". Afterwards, on June 13, 2010 it was announced that the system would officially be called Kinect, a blend of the words "kinetic" and "connect", which describe key aspects of the initiative. On November 4, 2010, Kinect was launched in North America, while in Europe on November 10, 2010.

Kinect holds the Guinness World Record of being "the fastest selling consumer electronics device", after selling a total of 8 million units in its first 60 days, from 4 November 2010 to 3 January 2011. GazDeaves, gaming editor for Guinness World Records, said that, "According to independent research, no other consumer electronics device sold faster within a 60-day time span, which is an incredible achievement considering the strength of the sector".

What it is revolutionary about Kinect is that it’s the world’s first project to combine full-body 3D motion capture, facial and voice recognition, with particular software, all in one device. The actual combination of hardware and software leads to a new way to control and interact. It is no need to hold any peripherals (no buttons, no remotes, and no joysticks); you just need to stand in front of the Kinect device and to use your body and natural movements, like speech and gestures.

Kinect builds on software technology developed internally by Rare, a subsidiary of Microsoft Game Studios owned by Microsoft, and on range camera technology by Israeli developer Prime Sense, which developed a system that can interpret specific gestures, making completely hands-free control of electronic devices possible by using an infrared projector and camera and a special microchip to track the movement of objects and individuals in three dimension. This 3D scanner system called Light Coding employs a variant of image-based 3D reconstruction.

The Kinect sensor is a horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device features an "RGB camera, depth sensor and multi-array microphone running proprietary software", which provide full-body 3D motion capture, facial recognition and voice recognition capabilities.

The depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions. The sensing range of the depth sensor is adjustable, and the Kinect software is capable of automatically calibrating the sensor based on gameplay and the player's physical environment, accommodating for the presence of furniture or other obstacles.

The software technology enables advanced gesture recognition, facial recognition and voice recognition. Kinect is capable of simultaneously tracking up to six people, including two active players for motion analysis with a feature extraction of 20 joints per player.

Kinect's various sensors output video at a frame rate of ~9 Hz to 30 Hz depending on resolution. The default RGB video stream uses 8-bit VGA resolution (640 × 480 pixels) with a Bayer color filter, but the hardware is capable of resolutions up to 1280x1024 (at a lower frame rate) and other color formats such as UYVY. The monochrome depth sensing video stream is in VGA resolution (640 × 480 pixels) with 11-bit depth, which provides 2,048 levels of sensitivity. The Kinect can also stream the view from its IR camera directly (i.e.: before it has been converting into a depth map) as 640x480 video, or 1280x1024 at a lower frame rate. The sensor has an angular field of view of 57° horizontally and 43° vertically, while the motorized pivot is capable of tilting the sensor up to 27° either up or down.

4.1.1 Sensor

Inside the sensor case, a Kinect for Windows sensor contains:

- An RGB camera that stores three channel data in a 1280x960 resolution. This makes capturing a color image possible.
- An infrared (IR) emitter and an IR depth sensor. The emitter emits infrared light beams and the depth sensor reads the IR beams reflected back to the sensor. The reflected beams are converted into depth information measuring the distance between an object and the sensor. This makes capturing a depth image possible.
- A multi-array microphone, which contains four microphones for capturing sound. Because there are four microphones, it is possible to record audio as well as find the location of the sound source and the direction of the audio wave.
- A 3-axis accelerometer configured for a 2G range, where G is the acceleration due to gravity. It is possible to use the accelerometer to determine the current orientation of the Kinect.

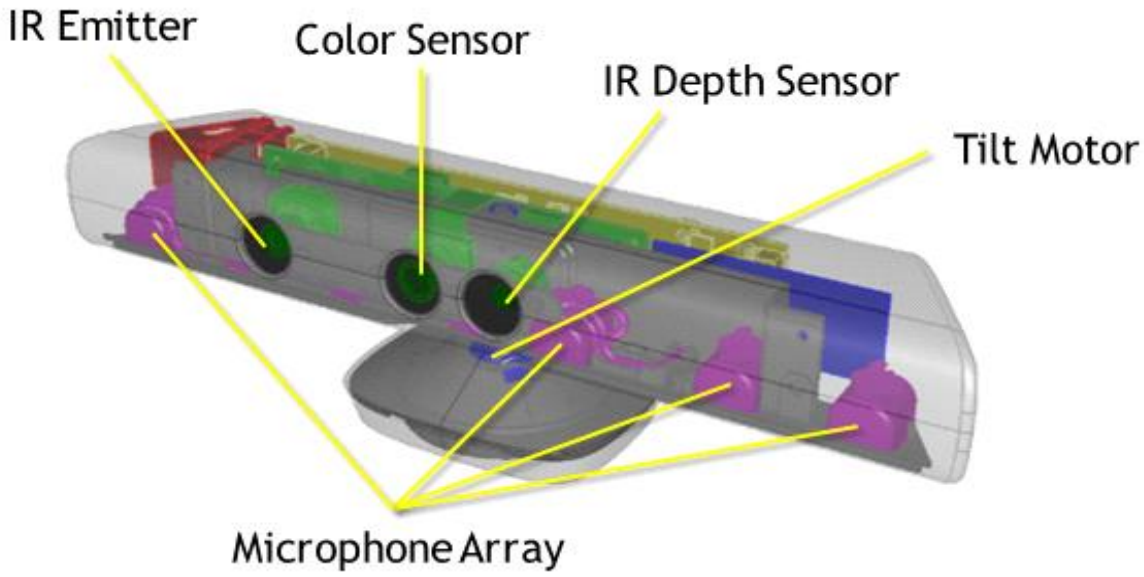


Figure 4.1. Internal Design of the Kinect

4.1.2 Kinect for Windows

On February 21, 2011 Microsoft announced that it would release a non-commercial Kinect software development kit (SDK) for Windows in spring 2011, which was released for Windows 7 on June 16, 2011 in 12 countries.

The SDK includes Windows 7 compatible PC drivers for Kinect device. It provides Kinect capabilities to developers to build applications with C++, C#, or Visual Basic by using Microsoft Visual Studio 2010 and includes following features:

Raw sensor streams: Access to low-level streams from the depth sensor, color camera sensor, and four-element microphone array.

Skeletal tracking: The capability to track the skeleton image of one or two people moving within the Kinect field of view for gesture-driven applications.

Advanced audio capabilities: Audio processing capabilities include sophisticated acoustic noise suppression and echo cancellation, beam formation to identify the current sound source, and integration with the Windows speech recognition API.

Figure 4.2. Kinect for Windows

4.1.3 Kinect SDK

The SDK provides the tools and APIs, both native and managed, that you need to develop Kinect-enabled applications for Microsoft Windows. Developing Kinect-enabled applications is essentially the same as developing other Windows applications, except that the Kinect SDK provides support for the features of the Kinect, including color images, depth images, audio input, and skeletal data.

Some examples of the types of Windows applications you can build using the functionality supported in this SDK:

- Recognize and track moving people using skeletal tracking.
- Determine the distance between an object and the sensor camera using depth data.
- Capture audio using noise and echo cancellation or find the location of the audio source.
- Enable voice activated applications by programming a grammar for use with a speech recognition engine.

The SDK includes:

- Drivers and technical documentation for implementing Kinect-enabled applications using a Kinect for Windows sensor.
- Reference APIs and documentation for programming in managed and unmanaged code. The APIs deliver multiple media streams with minimal software latency across various video, CPU, and device variables.
- Samples that demonstrate good practices for using a Kinect sensor.
- Example code that breaks down the samples into user tasks.

4.1.4 Kinect for Windows Architecture

The SDK provides a sophisticated software library and tools to help developers use the rich form of Kinect-based natural input, which senses and reacts to real-world events. The Kinect and the software library interact with your application, as shown in Figure 1.

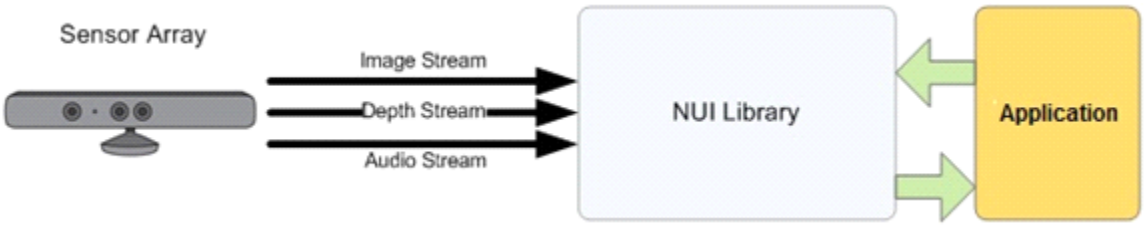


Figure 4.3. Hardware and Software Interaction with an Application

The components of the SDK include:

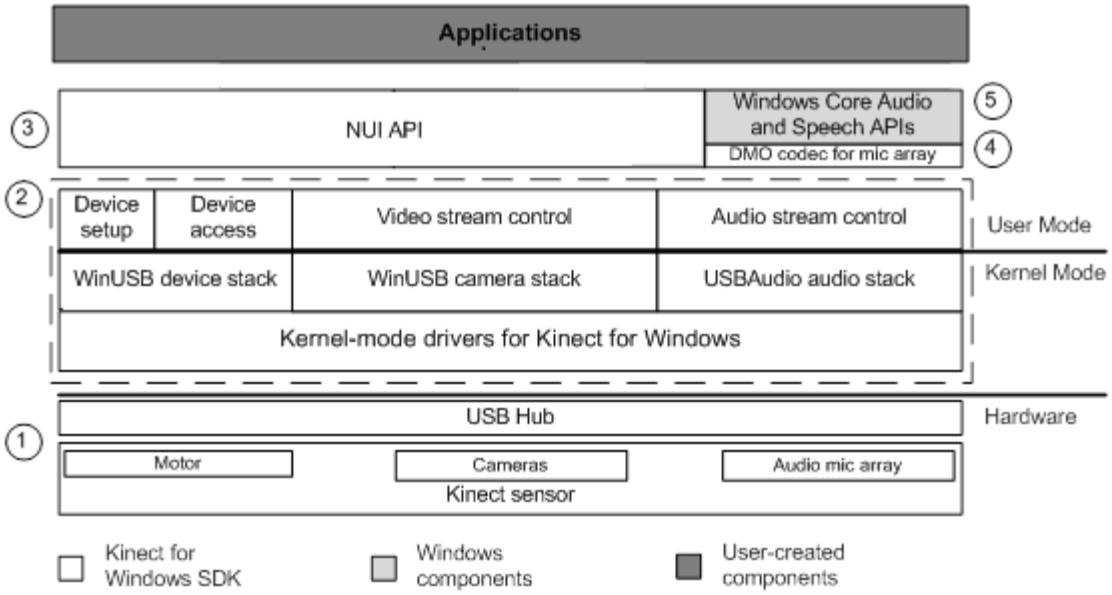


Figure 4.4. SDK Architecture

These components include the following:

Kinect hardware - The hardware components, including the Kinect sensor and the USB hub through which the Kinect sensor is connected to the computer.

Kinect drivers - The Windows drivers for the Kinect, which are installed as part of the SDK setup process as described in this document. The Kinect drivers support:

The ***Kinect microphone array*** as a kernel-mode audio device that you can access through the standard audio APIs in Windows.

Audio and video streaming controls for streaming audio and video (color, depth, and skeleton).

Device enumeration functions that enable an application to use more than one Kinect.

Audio and Video Components

Kinect natural user interface for skeleton tracking, audio, and color and depth imaging

DirectX Media Object (DMO) for microphone array beamforming and audio source localization.

Windows 7 standard APIs - The audio, speech, and media APIs in Windows 7, as described in the Windows 7 SDK and the Microsoft Speech SDK. These APIs are also available to desktop applications in Windows 8.

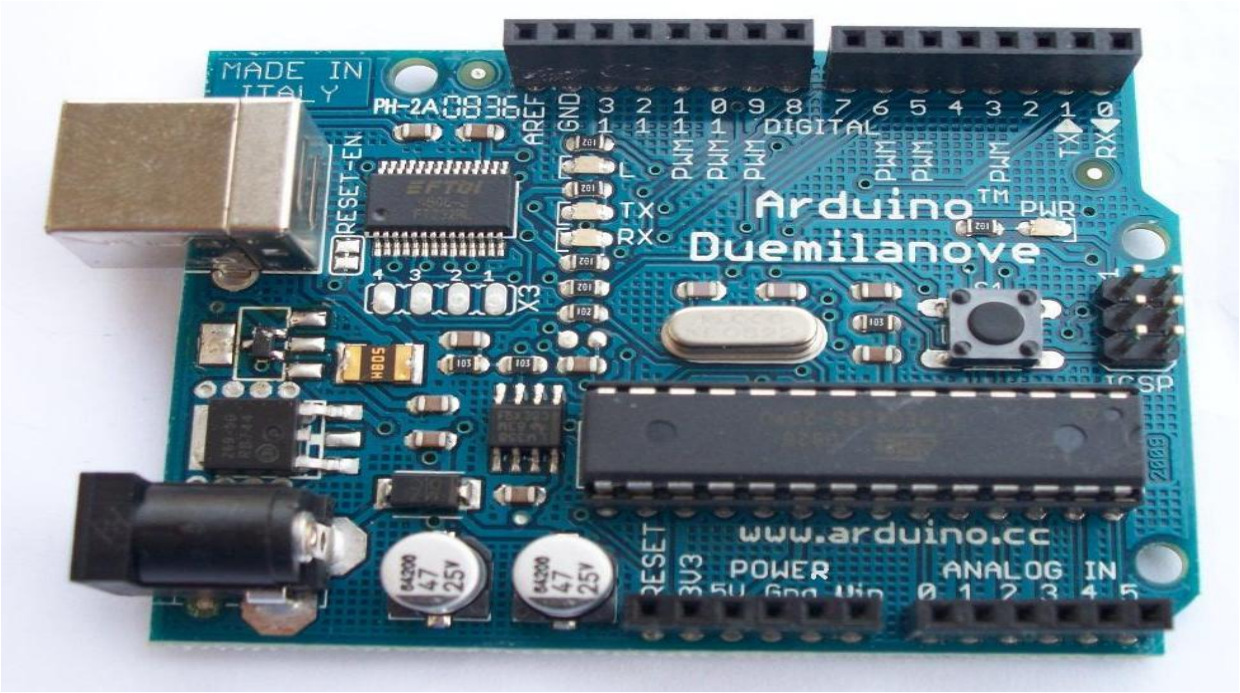
4.2 ARDUINO

Arduino is a single-board microcontroller designed to make the process of using electronics in multidisciplinary projects more accessible. The hardware consists of a simple open source hardware board designed around an 8-bit Atmel AVR microcontroller, though a new model has been designed around a 32-bit Atmel ARM. The software consists of a standard programming language compiler and a boot loader that executes on the microcontroller.

Arduino boards can be purchased pre-assembled or do-it-yourself kits. Hardware design information is available for those who would like to assemble an Arduino by hand. There are sixteen official Arduinos that have been commercially produced to date. Numerous hardware variations of the Arduino are being sold by third parties.

4.2.1 Hardware

An Arduino board consists of an Atmel 8-bit AVR microcontroller with complementary components to facilitate programming and incorporation into other circuits. An important aspect of the Arduino is the standard way that connectors are exposed, allowing the CPU board to be connected to a variety of interchangeable add-on modules known as shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an I²C serial bus, allowing many shields to be stacked and used in parallel. Official Arduinos have used the mega AVR series of chips, specifically the ATmega8, ATmega168, ATmega328, ATmega1280, and ATmega2560. A handful of other processors have been used by Arduino compatibles. Most boards include a 5 volt linear regulator and a 16 MHz crystal oscillator (or ceramic resonator in some variants). An Arduino's microcontroller is also pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory, compared with other devices that typically need an external programmer.



4.5 Arduino Board

4.2.2 Software

The Arduino integrated development environment (IDE) is a cross-platform application written in Java, and is derived from the IDE for the Processing programming language and the Wiring projects. It is designed to introduce programming to artists and other newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. There is typically no need to edit make files or run programs on a command-line interface.

Arduino programs are written in C or C++. The Arduino IDE comes with a software library called "Wiring" from the original Wiring project, which makes many common input/output operations much easier. Users only need define two functions to make a runnable cyclic executive program:

- setup(): a function run once at the start of a program that can initialize settings
- loop(): a function called repeatedly until the board powers off

It is a feature of most Arduino boards that they have an LED and load resistor connected between pin 13 and ground, a convenient feature for many simple tests. The previous code would not be seen by a standard C++ compiler as a valid program, so when the user clicks the "Upload to I/O board" button in the IDE, a copy of the code is written to a temporary file with an extra include header at the top and a very simple main() function at the bottom, to make it a valid C++ program.

The Arduino IDE uses the GNU toolchain and AVR Libc to compile programs, and uses avrdude to upload programs to the board.

5. DESIGN DETAILS

5.1 SKELETAL TRACKING

Skeletal Tracking allows Kinect to recognize people and follow their actions.Using the infrared (IR) camera, Kinect can recognize up to six users in the field of view of the sensor. Of these, up to two users can be tracked in detail. An application can locate the joints of the tracked users in space and track their movements over time.

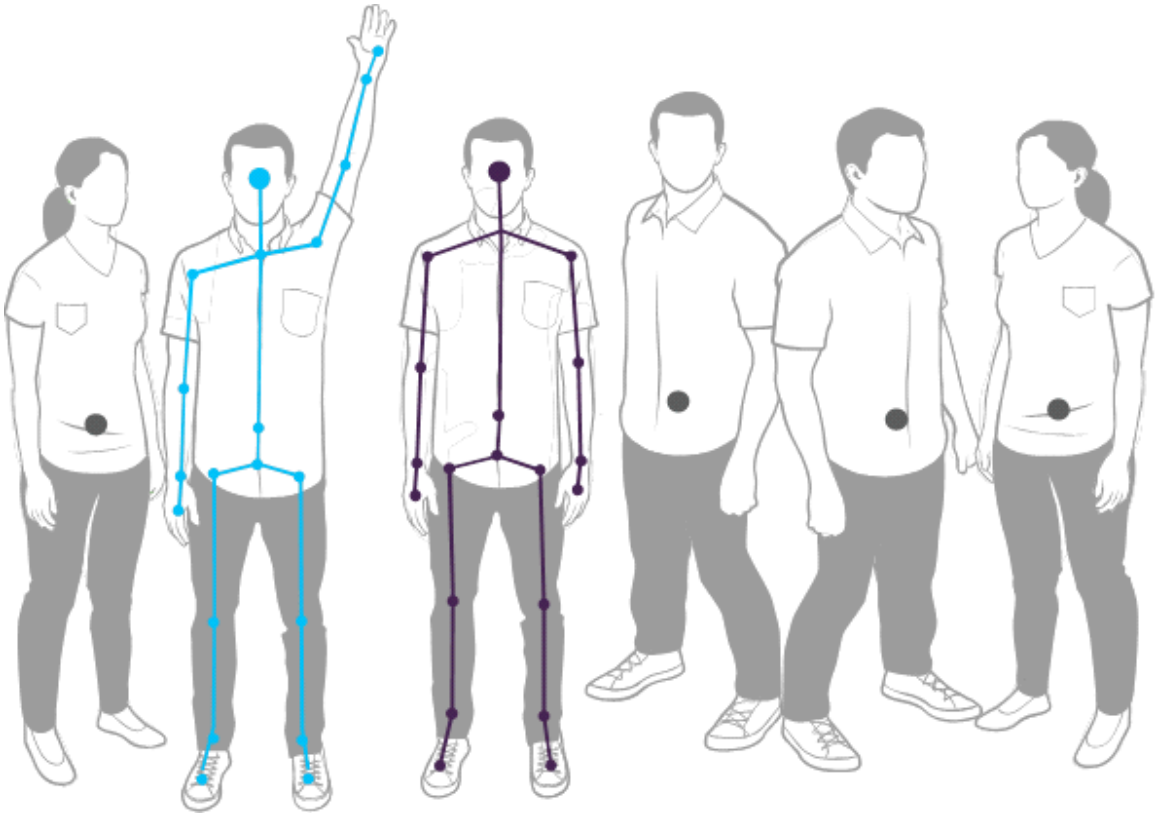


Figure 5.1. Kinect can recognize six people and track two

Skeletal Tracking is optimized to recognize users standing or sitting, and facing the Kinect; sideways poses provide some challenges regarding the part of the user that is not visible to the sensor.

To be recognized, users simply need to be in front of the sensor, making sure the sensor can see their head and upper body; no specific pose or calibration action needs to be taken for a user to be tracked.

5.1.1 Field of View

Kinect field of view of the users is determined by the settings of the IR camera, which are set with the DepthRange Enumeration.

In default range mode, Kinect can see people standing between 0.8 meters (2.6 feet) and 4.0 meters (13.1 feet) away; users will have to be able to use their arms at that distance, suggesting a practical range of 1.2 to 3.5 meters. For more details, see the k4w_hig_main.

5.1.2 Skeletal Tracking Precision and Multiple Kinect Sensors

The infrared emitter of a Kinect sensor projects a pattern of infrared light. This pattern of light is used to calculate the depth of the people in the field of view allowing the recognition of different people and different body parts. If you use more than one Kinect sensor to illuminate the target area, you may notice a reduction in the accuracy and precision of skeletal tracking due to interference with the infrared light sources. To reduce the possibility of interference, it is recommended that no more than one Kinect sensor (or infrared light source) points to a field of view where skeletal tracking is being done.

In near range mode, Kinect can see people standing between 0.4 meters (1.3 feet) and 3.0 meters (9.8 feet); it has a practical range of 0.8 to 2.5 meters.

Skeleton tracking a new modality, call seated mode for tracking user skeletons.

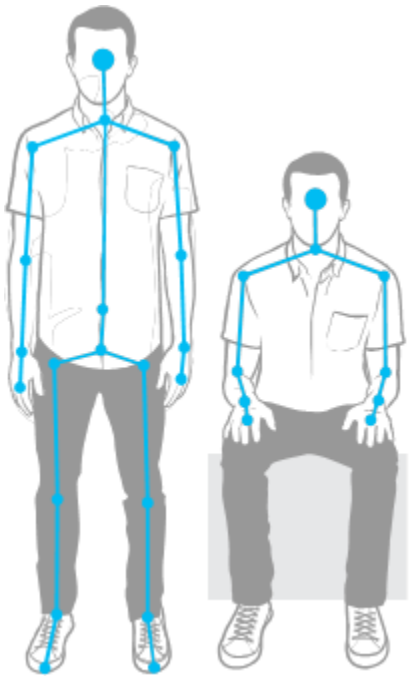


Figure 5.2. Skeletal tracking Default and Near Mode

The seated tracking mode is designed to track people who are seated on a chair or couch, or whose lower body is not entirely visible to the sensor. The default tracking mode, in contrast, is optimized to recognize and track people who are standing and fully visible to the sensor.

5.2 ROBOTIC ARM

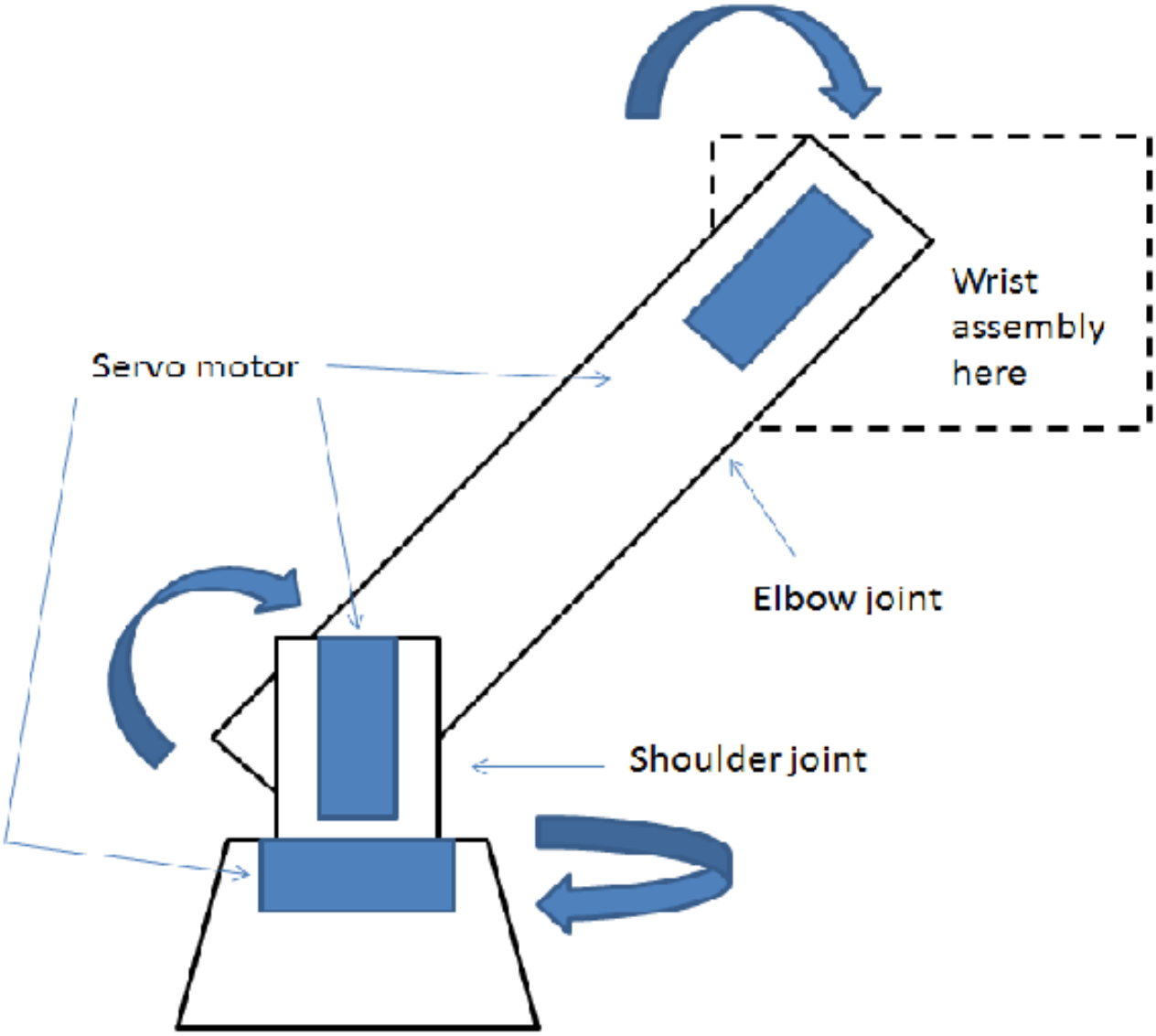


Figure 5.3. Arm Assembly

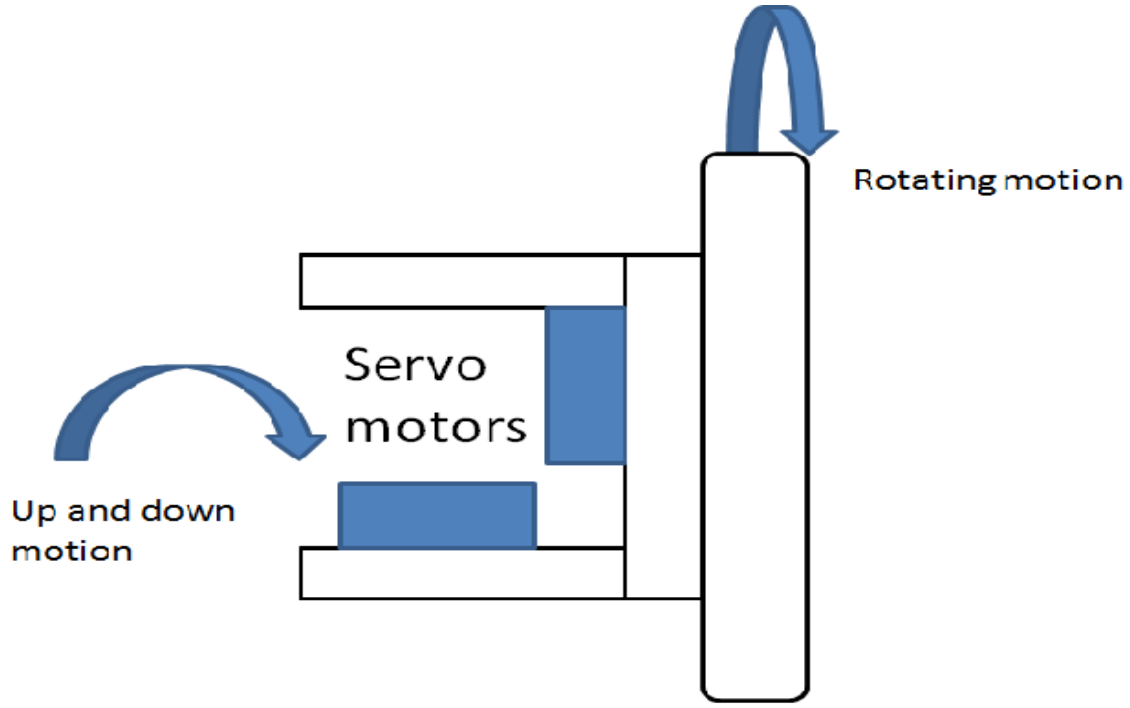


Figure 5.4. Wrist Assembly

6. IMPLEMENTATION

6.1 KINECT IMPLEMENTATION

6.1.1 Code to check if the sensor is connected

This portion of the code looks for a connected sensor and turns on the skeleton stream to receive skeleton frames.

```
private void WindowLoaded(object sender, RoutedEventArgs e)
    {
        this.drawingGroup = new DrawingGroup();
        this.imageSource = new DrawingImage(this.drawingGroup);
        Image.Source = this.imageSource;
        foreach (var potentialSensor in KinectSensor.KinectSensors)
        {
            if (potentialSensor.Status == KinectStatus.Connected)
            {
                this.sensor = potentialSensor;
                break;
            }
        }
        if (null != this.sensor)
        {
            this.sensor.SkeletonStream.Enable();
            this.sensor.SkeletonFrameReady += this.SensorSkeletonFrameReady;
            try
            {
                this.sensor.Start();
            }
```

```
        catch (IOException)
        {
            this.sensor = null;
        }
    }
```

6.1.2 Code to obtain the angle between two joints

This portion of the code defines the joints, defines vectors for the same, obtains angle between them and normalizes the values.

```
public void ToGetTheAngleBetweenJoints(Skeleton skeleton)
    {
        Joint r_wrist = skeleton.Joints[JointType.WristRight];
        Joint r_shoulder = skeleton.Joints[JointType.ShoulderRight];
        Joint r_elbow = skeleton.Joints[JointType.ElbowRight];
        Joint r_hand = skeleton.Joints[JointType.HandRight];
        double x1 = r_shoulder.Position.X;
        double y1 = r_shoulder.Position.Y;
        double z1 = r_shoulder.Position.Z;
        double x2 = r_elbow.Position.X;
        double y2 = r_elbow.Position.Y;
        double z2 = r_elbow.Position.Z;
        double x3 = r_wrist.Position.X;
        double y3 = r_wrist.Position.Y;
        double z3 = r_wrist.Position.Z;
        double x4 = r_hand.Position.X;
        double y4 = r_hand.Position.Y;
        double z4 = r_hand.Position.Z;
```

```
            Vector3D v1 = new Vector3D(x2,y2,z2);
            Vector3D v2 = new Vector3D(x1,y1,z1);
            Vector3D v3 = new Vector3D(x3,y3,z3);
            Vector3D v4 = new Vector3D(x3, y3, z3);
            Vector3D v5 = new Vector3D(x2, y2, z2);
            Vector3D v6 = new Vector3D(x4, y4, z4);
            Vector3D b1 = v3 - v1;
            Vector3D b2 = v2 - v1;
            Vector3D b3 = v6 - v4;
            Vector3D b4 = v5 - v4;
            b1.Normalize();
            b2.Normalize();
            b3.Normalize();
            b4.Normalize();
            double AngleInRadiansForElbowMovement = AngleBetweenTwoVectors(b1, b2);
            double AngleInDegForElbowMovement = AngleInRadiansForElbowMovement *
(180 / Math.PI);
            //Angle.Content = (int)AngleInDegForElbowMovement;
            double AngleInRadiansForWristMovement = AngleBetweenTwoVectors(b3, b4);
            double AngleInDegForWristMovement = AngleInRadiansForWristMovement * (180
/ Math.PI);
            Angle.Content = (int)AngleInDegForWristMovement;
        }
```

6.1.3 Code to calculate angle between two vectors

This portion of the code calculates the angle between two vectors.

```
public float AngleBetweenTwoVectors(Vector3D vectorA, Vector3D vectorB)
        {
           double dotProduct = 0.0f;
           dotProduct=  Vector3D.DotProduct(vectorA,  vectorB);
           return (float)Math.Acos(dotProduct);
        }
```

6.2 ARDUINO IMPLEMENTATION

This portion of the code is to receive serial data from the Kinect and control the hardware component, i.e. the robot itself.

```
#include <Servo.h>
Servo elbow1,elbow2,wrist,gripper;
unsigned long intnum=0,value;
intpos,digit,mode;
char ch;
void setup()
{
Serial.begin(9600);
elbow1.attach(10);
elbow2.attach(11);
wrist.attach(12);
gripper.attach(13);
}
void loop()
{
if(Serial.available())
  {
delay(5);
switch(Serial.read())
     {
case 'A': Motors(1);
break;
```

```
case 'B':Motors(2);
break;
case 'G':gripper.write(00);
break;
case 'U':gripper.write(90);
break;
case 'P': moveForward();
delay(1000);
break;
case 'R': turnLeft();
delay(1000);
break;
case 'L':
turnRight();
delay(1000);
break;
case 'S':stop();
delay(1000);
break;
    }
  }
}
void Motors(intmotor_num)
{
ch=Serial.read();
while(ch!='E')
   {
digit=(ch-48);
num=(num*10)+digit;
delay(5);
ch=Serial.read();
    }
switch(motor_num)
  {
case 1:elbow1.write(num);elbow2.write(num);
break;
case 2:wrist.write(num);
break;
   }
num=0;
}
voidmoveForward()
{
digitalWrite(2,HIGH);
digitalWrite(3,LOW);
digitalWrite(4,HIGH);
```

```
digitalWrite(5,LOW);
digitalWrite(6,HIGH);
digitalWrite(7,LOW);
digitalWrite(8,HIGH);
digitalWrite(9,LOW);
}
voidturnRight()
{
digitalWrite(2,LOW);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,LOW);
digitalWrite(6,HIGH);
digitalWrite(7,LOW);
digitalWrite(8,LOW);
digitalWrite(9,HIGH);
}
voidturnLeft()
{
digitalWrite(2,HIGH);
digitalWrite(3,LOW);
digitalWrite(4,LOW);
digitalWrite(5,HIGH);
digitalWrite(6,LOW);
digitalWrite(7,HIGH);
digitalWrite(8,HIGH);
digitalWrite(9,LOW);
}
void stop()
{
digitalWrite(2,HIGH);
digitalWrite(3,HIGH);
digitalWrite(4,HIGH);
digitalWrite(5,HIGH);
digitalWrite(6,HIGH);
digitalWrite(7,HIGH);
digitalWrite(8,HIGH);
digitalWrite(9,HIGH);
}
```

7. TEST CASES

7.1 SOFTWARE TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs (errors or other defects).

Software testing can be stated as the process of validating and verifying that a computer program/application/product:

- meets the requirements that guided its design and development,
- works as expected,
- can be implemented with the same characteristics,
- satisfies the needs of stakeholders.

Software Testing is the process of executing a program or system with the intent of finding errors. Or, it involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Software is not unlike other physical processes where inputs are received and outputs are produced. Where software differs is in the manner in which it fails. Most physical systems fail in a fixed (and reasonably small) set of ways. By contrast, software can fail in many bizarre ways. Detecting all of the different failure modes for software is generally infeasible.

7.2 TEST PLAN

Testing is carried out in multiple levels. Activities at each level must be planned well in advance and it has to be formally documented. Based on the individual plan only, the individual test levels are carried out. Testing process starts with a test plan that identifies all the testing related activities that must be performed and specifies the schedule, allocates the resources and specifies guidelines for testing. The testing done for this project is unit testing.

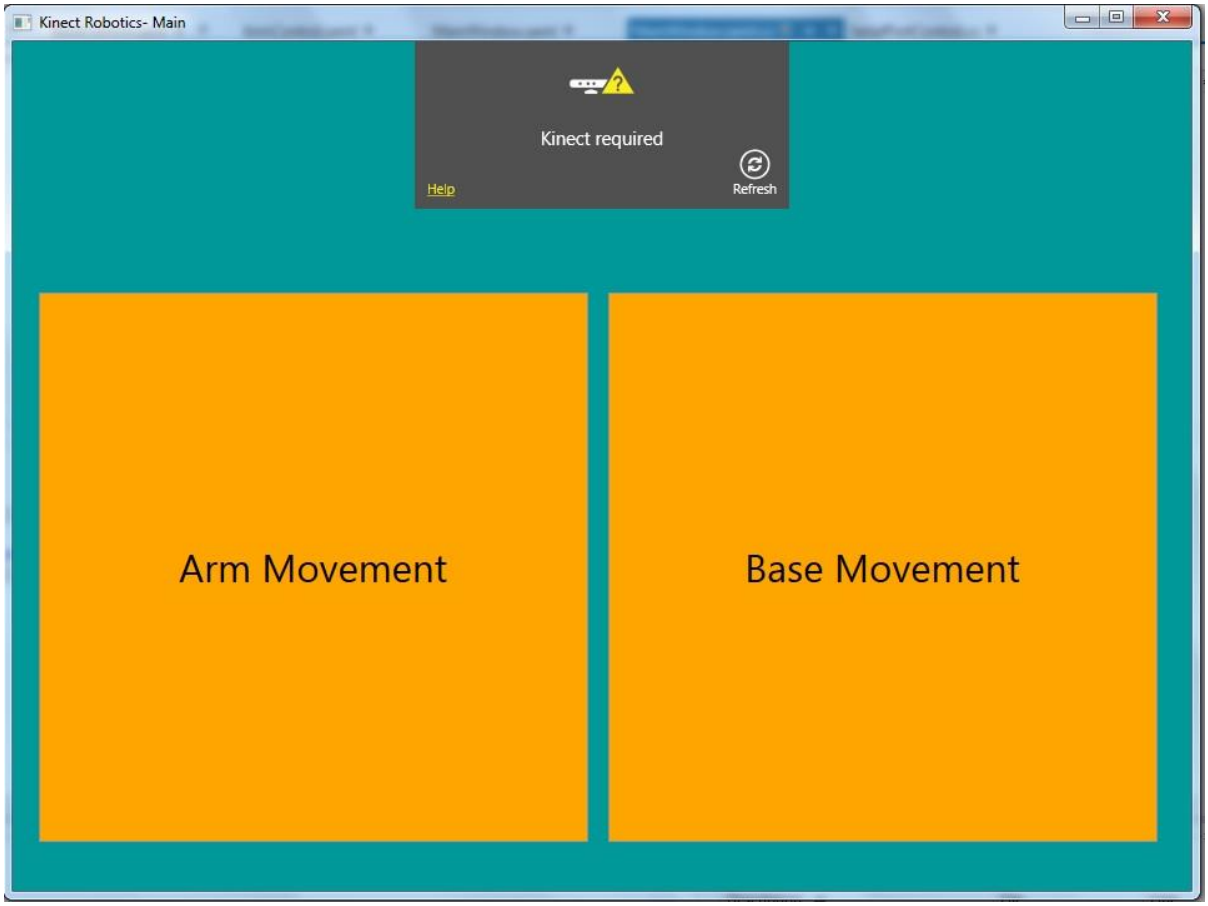
7.2.1 Unit Testing

A unit test is a method of testing the correctness of a particular module of source code. The idea is to write test cases for every no-trivial function or method in the module so that each test case is separate from the others if possible. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.

In this project we test the connectivity and the working of the Kinect, working of the Arduino board, serial port communication in separate modules.

7.3 TEST CASES

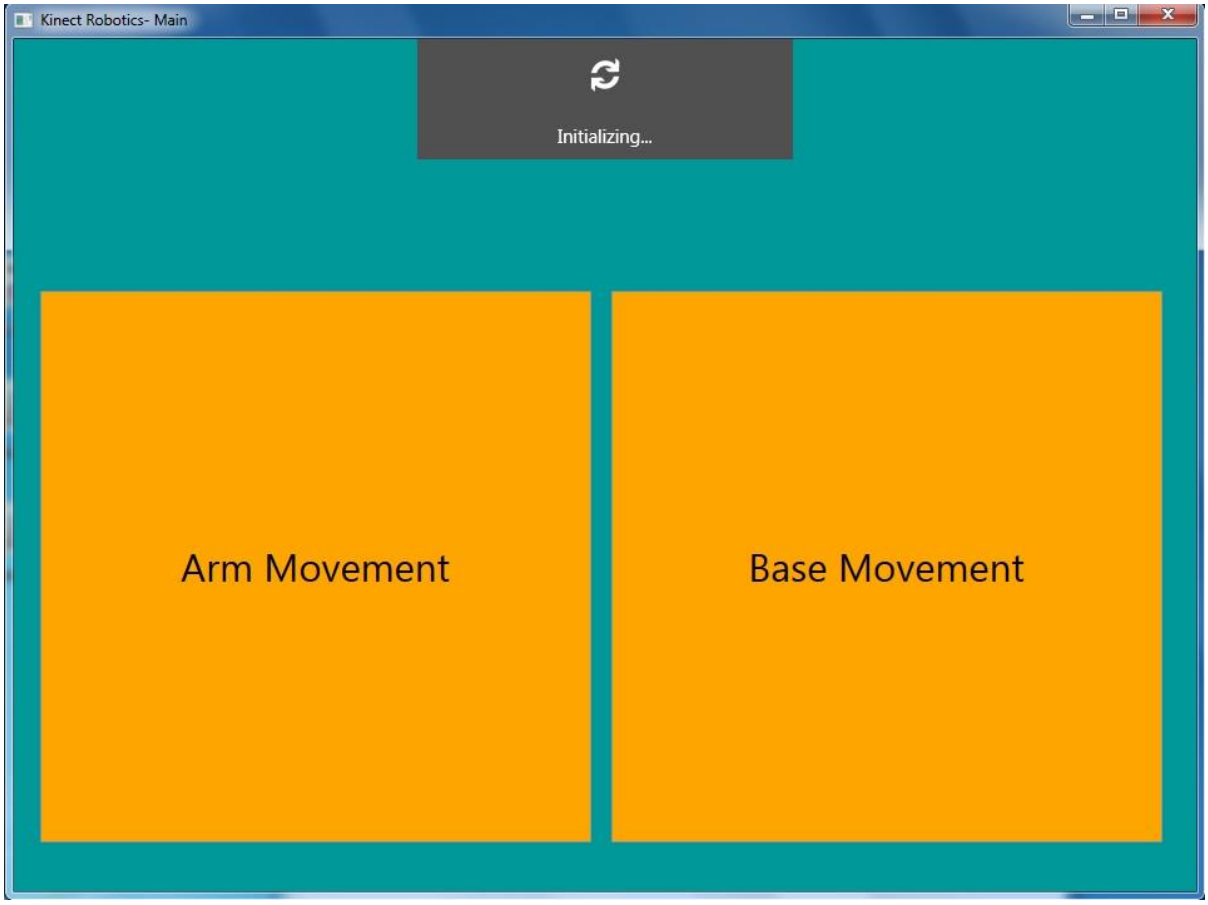
7.3.1 To check if the sensor is connected




```
if ((this.Status & ChooserStatus.NoAvailableSensors) != 0)
 {
            newVisualState = "NoAvailableSensors";
            message = Resources.MessageNoAvailableSensors;
  }
```

```
//xaml code
<data name="MessageNoAvailableSensors" xml:space="preserve">
<value>Kinect required</value>
 </data>
```

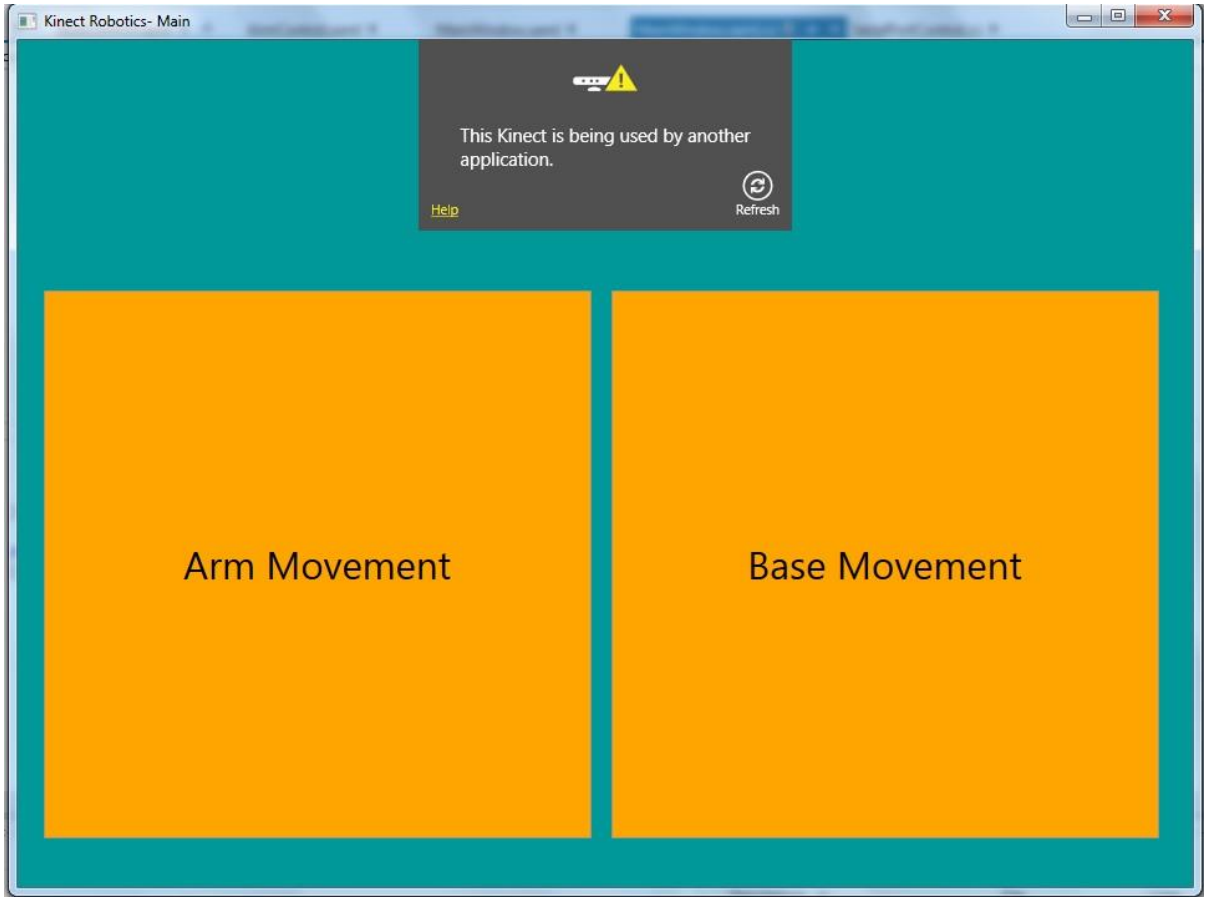
7.3.2 To check if the sensor is initializing




```
if ((this.Status & ChooserStatus.SensorInitializing) != 0)
{
            newVisualState = "Initializing";
            message = Resources.MessageInitializing;
  }

//xaml code
<data name="MessageInitializing" xml:space="preserve">
    <value>Initializing...</value>
  </data>
```

7.3.3 To check if the sensor is being used by another application




```
if ((this.Status & ChooserStatus.NoAvailableSensors) != 0)
{
        newVisualState = "NoAvailableSensors";
        message = Resources.MessageNoAvailableSensors;
}

//xaml code
<data name="MessageNoAvailableSensors" xml:space="preserve">
    <value>Kinect required</value>
  </data>
```

7.3.4 To check if the COM port is connected

CONCLUSION

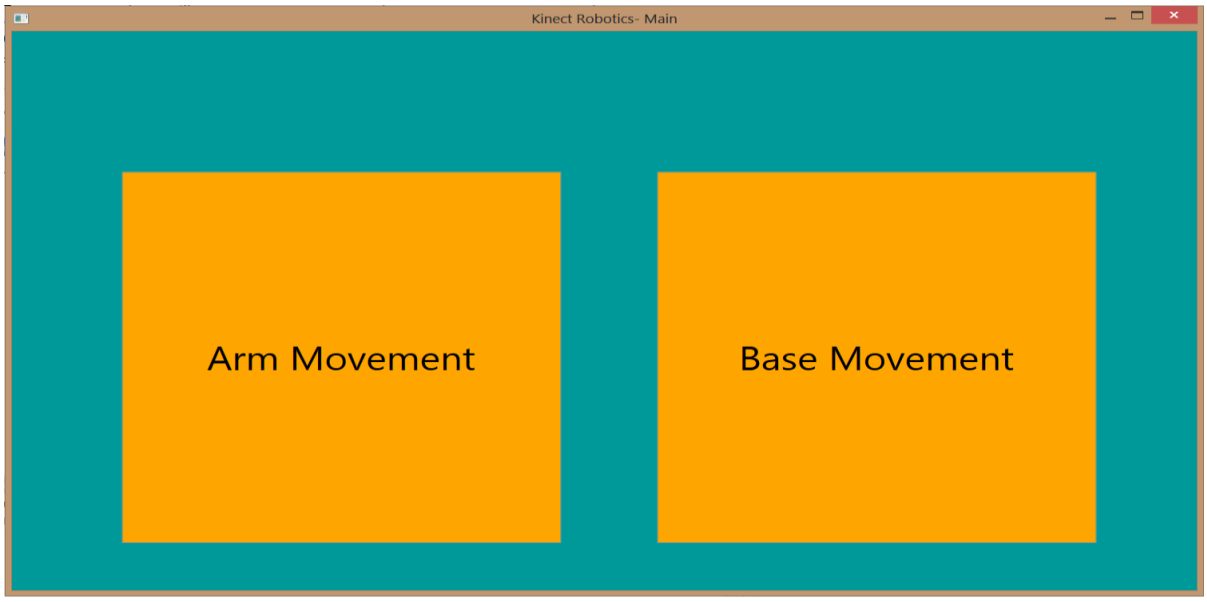
Microsoft Kinect for Windows– enabled applications open a broad range of new possibilities for people to interact with computers in ways that feel natural. Learning from demonstration is the scientific field which studies one of the easier ways a human has to deal with a humanoid robot: mimicking the particular task the subject wants to see reproduced by the robot. To achieve this a gesture recognition system is required.

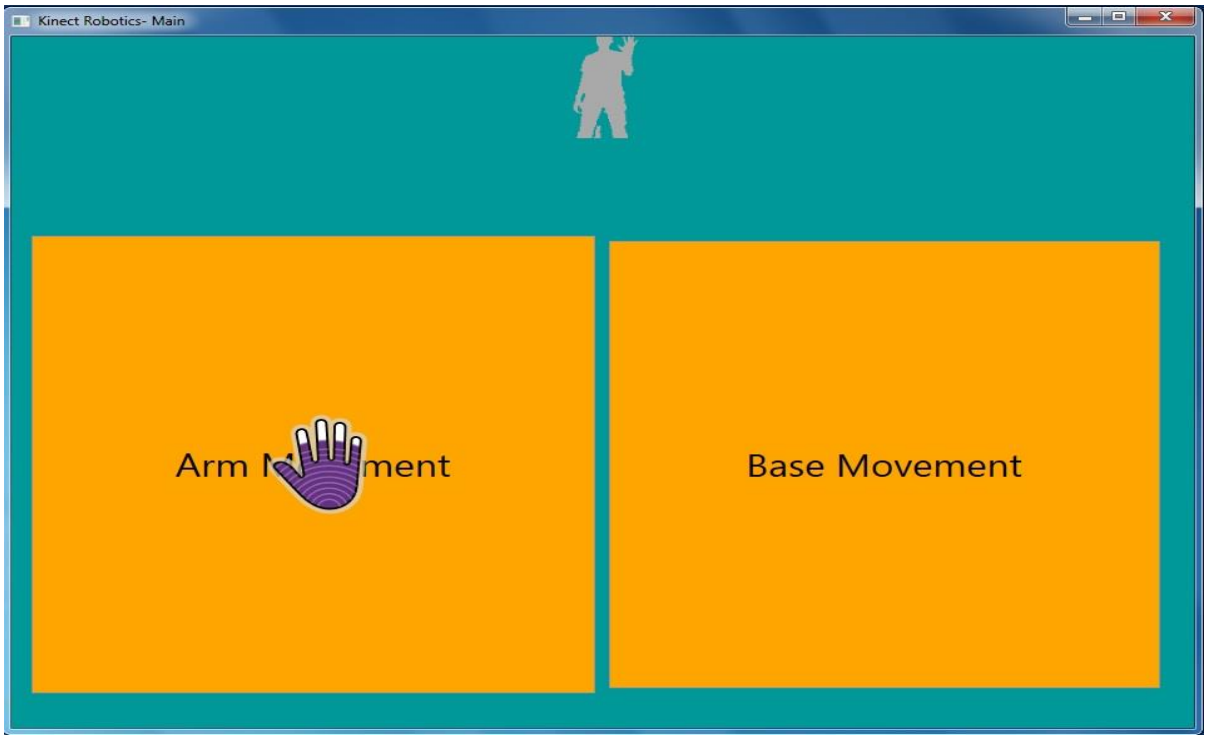
Here we present a novel and cheap humanoid robot implementation along with a visual, gesture-based interface, which enable users to deal with it. To catch and control subject's gestures we employed the Microsoft Kinect RGB-D. Users are allowed to control the robot just by mimicking the gestures they want to be performed by the robot in front of the depth camera.

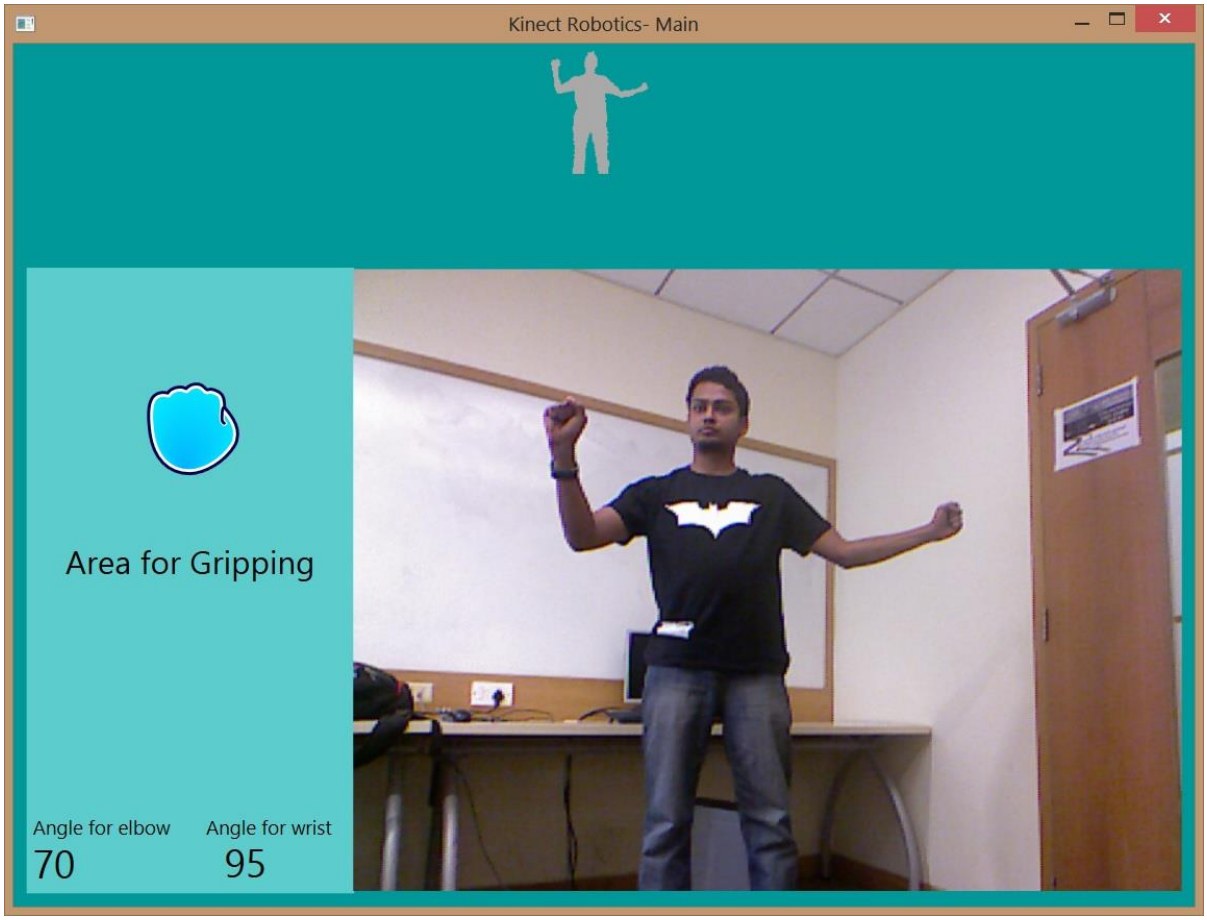
This should be seen as preliminary work, where we are providing elementary interaction tools, and should be extended in many different fashions, depending on the tasks the robot should perform. Few of the tasks that the robot could possibly perform is bomb defusal or cleaning hazardous environments where it is difficult for the humans to be physically present.

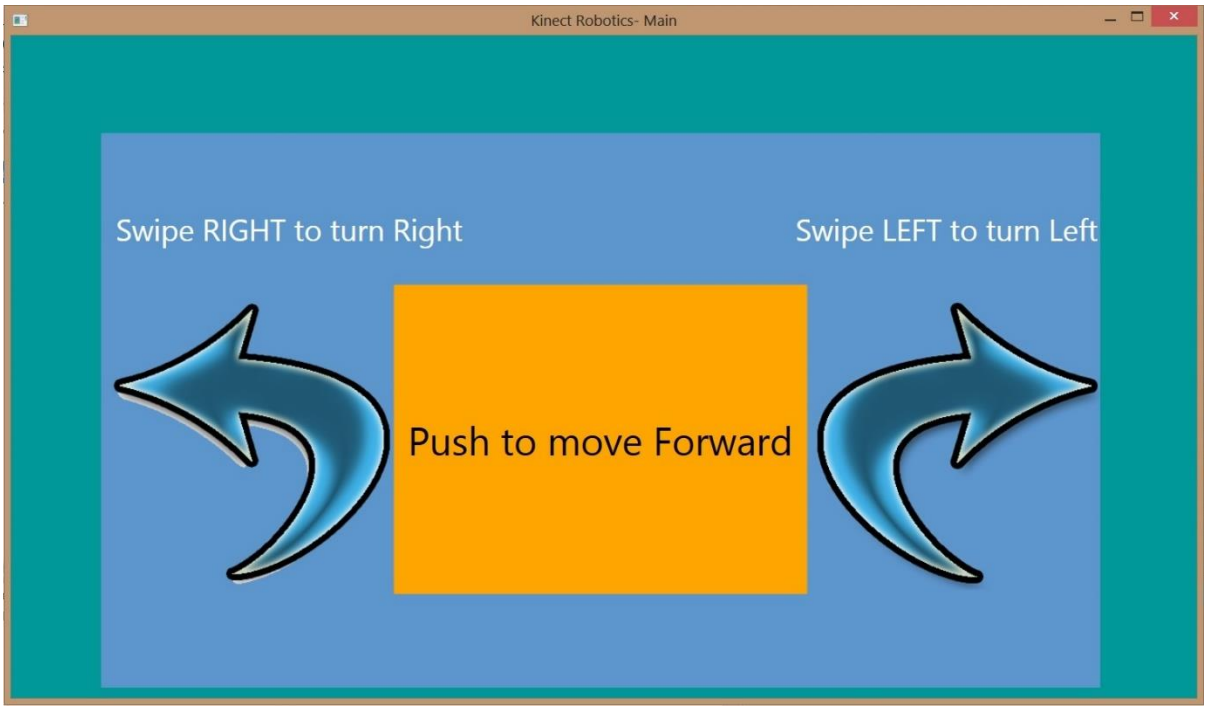
APPENDIX A

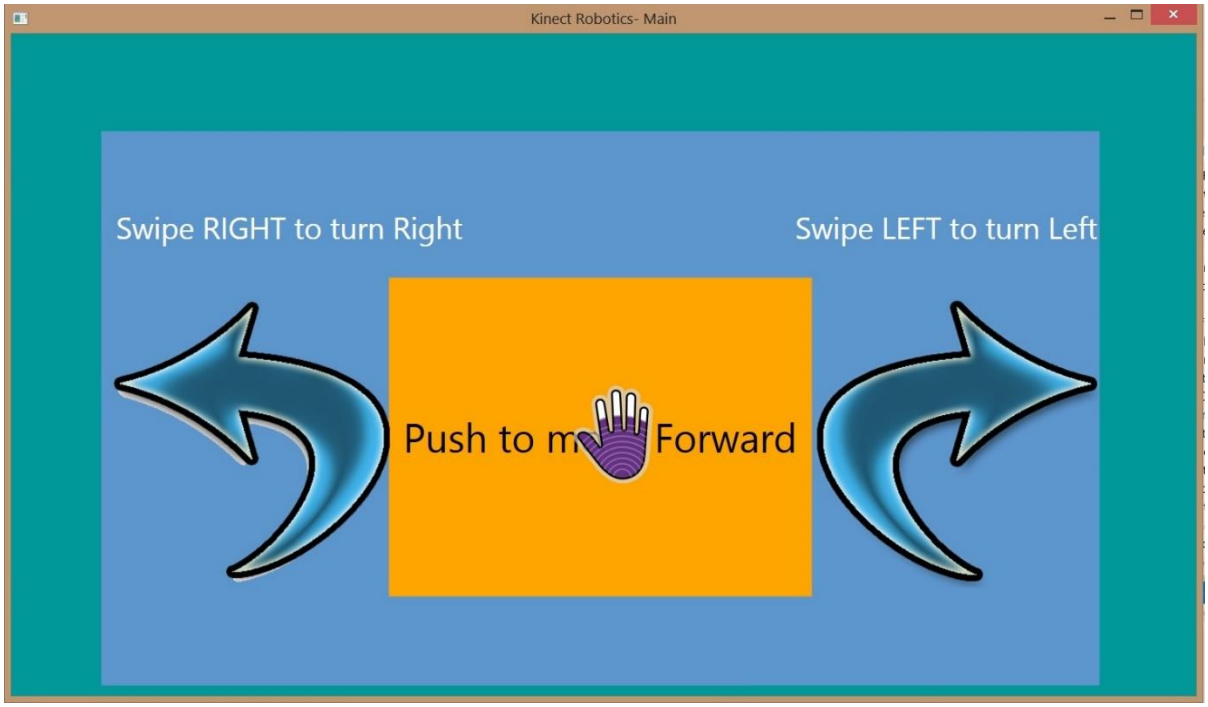
SNAPSHOTS

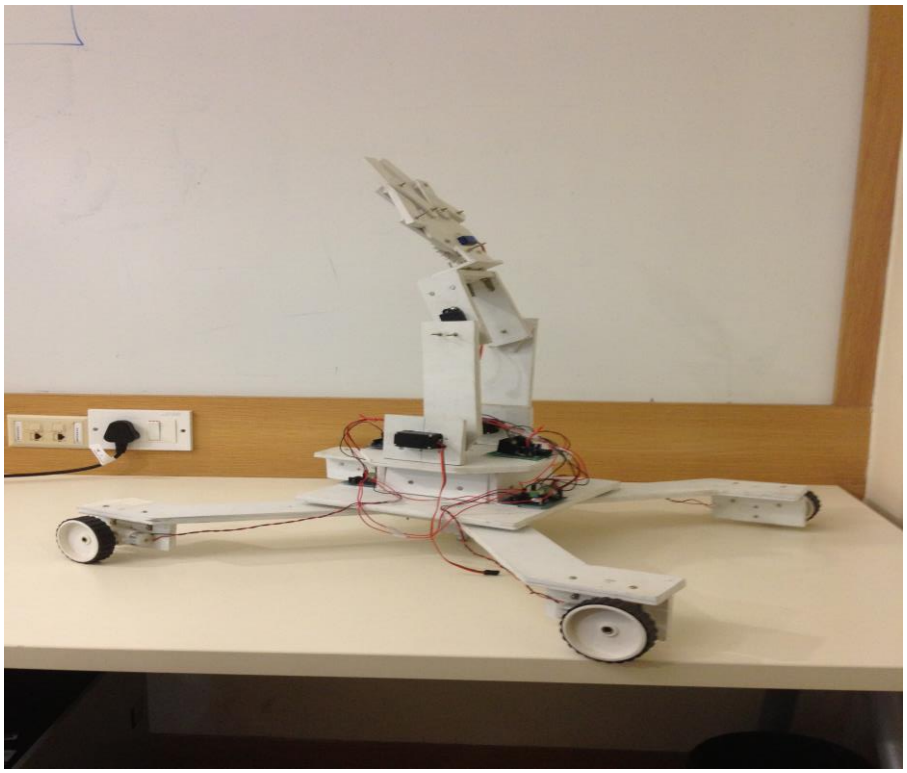


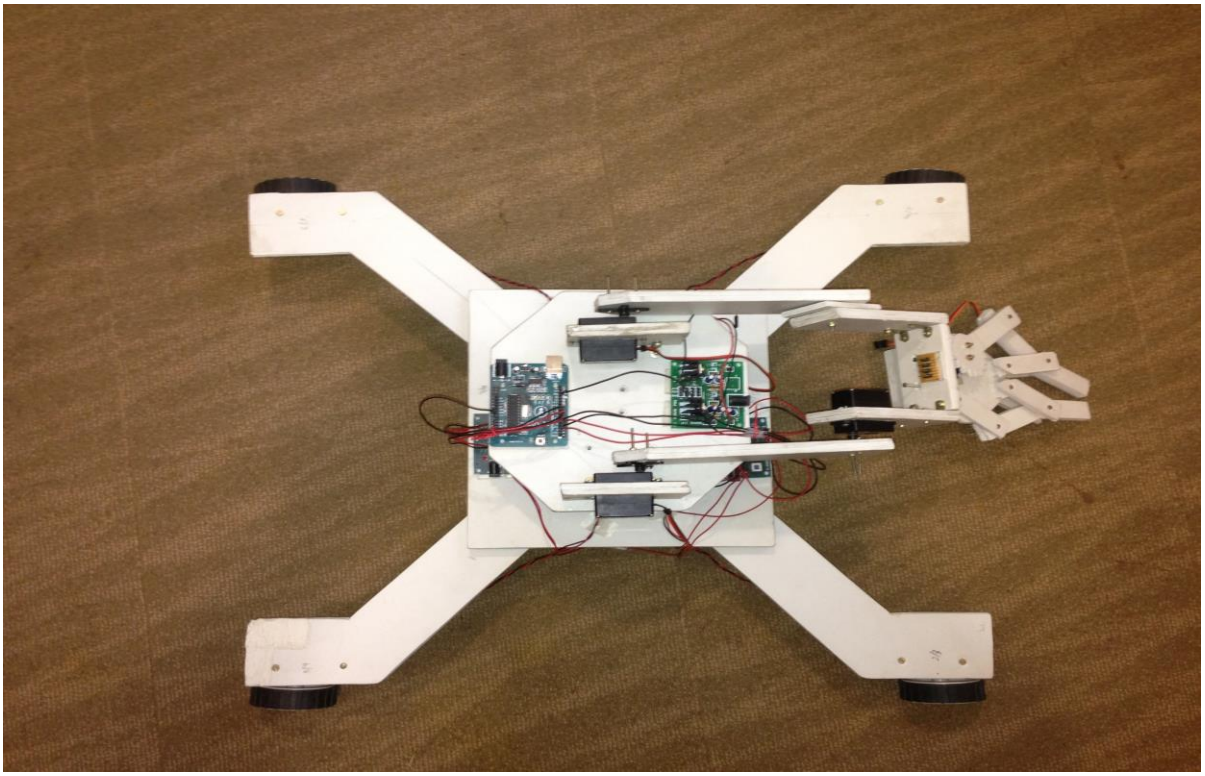



Kinect Robotics- Main
Area for Gripping
Angle for elbow
70
Angle for wrist
95


Kinect Robotics- Main
Swipe RIGHT to turn Right
Swipe LEFT to turn Left
Push to move Forward


Kinect Robotics- Main
Swipe RIGHT to turn Right
Swipe LEFT to turn Left
Push to m Forward





BIBLIOGRAPHY

[1]Jeff Kramer, Nicolas Burrus, Florian Echtler, Daniel Herrera C., and Matt Parker:"Hacking the Kinect"

[2] O'Reilly: "Kinect Hacks: Tips and Tools for Motion and Pattern Recognition"

[3] David Catuhe: "Microsoft Programming with the Kinect for Windows Software Development Kit"

[4] http://msdn.microsoft.com