Kinect Controlled Robotic Arm

System Requirements Specification

Aayush Singhal (aayush@cse) 09005041
Sivakanth Gopi (gopisivakanth@cse) 09005043
Sriram Bhargav Karnati (ram@cse) 09005067
Aakash Rao N S (aakash@cse) 09005069

Contents

1 Introduction

The purpose of this document is to present a detailed description of the Kinect Controlled Robotic Arm. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate. This document is intended for the developers of the system.

1.1 Definitions

Term	**Definition**
Robotic Arm	A programmable mechanical arm wih similar functions to a human arm
Arm	Unless specified otherwise, refers to the robotic arm
Kinect	A motion sensing input device developed by Microsoft
User	The person controlling the robotic arm
Depth Data	The data produced by the depth image camera in Kinect. Each frame in the stream contains the distance, in mm, to the nearest object at each x and y coordinate in the camera's field of view.
Skeletal Data	The data provided by Kinect about the configuration of a skeleton, as a set of positions that compose the sleleton. The skeleton represents the user's position and posture.
VGA	Video Graphics Accelerator
SDK	Software Development Kit
IDE	Integrated Development Environment
API	Application Programming Interface
GUI	Graphical User Interface
DirectX	A collection of API for handling multimedia related tasks, on Microsoft platforms
AVR Studio	IDE for C programming for the robotic arm
IR	Infra-Red
ZigBee	A specification for a suite of high level communication protocols using small, low-power digital radios based on an IEEE 802 standard for personal area networks
X-CTU	Software used to configure the Zigbee wireless module

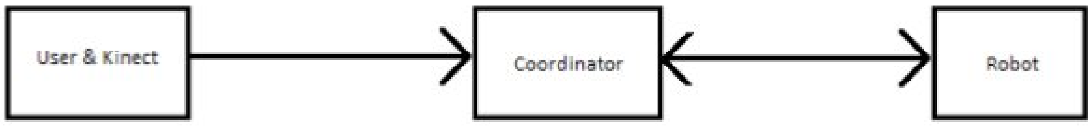
Figure 1.1: Microsoft Kinect

1.2 References

- Kinect for Windows SDK v1 : Resources and Documentation, Microsoft Corporation
  `http://www.microsoft.com/en-us/kinectforwindows/develop/resources.aspx`
- Kinect SDK v1 Quick Starts, Channel 9, MSDN
  `http://channel9.msdn.com/Series/KinectQuickstart`
- Dexter ER2 Robotic Arm User Manual, Nex Robotics, Version 1.00, April 2011
  `http://www.nex-robotics.com/products/robotic-arms-and-grippers/dexter-er-2-heavy-dut`
  `html`
- AVR AtMega2560 Programming Manual, Atmel, May 2011
  `http://www.atmel.com/Images/doc2549.pdf`

2 Overall Description

2.1 System Environment



Kinect Controlled Robotic Arm has 2 active systems and 2 coordinating systems. Hand which is to be imitated, and robotic arm are active systems. And computer and kinect are coordinating systems. User's hand is able to communicate with robotic arm via coordinating systems. The communication between robotic arm and coordinating system is on a dedicated wireless frequency channel. While kinect is connected to computer by a wired connection.

2.2 Product Perspective

Kinect captures the movement and gestures of hand and body using its various sensors. Robotic arm is controlled remotely using ZigBee wireless interface. Computer receives hand movements from the kinect sensor and translates it to a particular command and sends that command to Robotic arm. Robotic arm looks as if, it is imitating the hand movement.

In order to make the system completely remote control, a camera may be mounted on the top of robotic arm. So that user can view the environment of robotic arm from distance. Connection to Robotic arm can also be established without using ZigBee wireless device, and through a wired connection, because Robotic arm is kept stationary at all times.

2.3 Product Functions

2.3.1 Description

The user moves hands to define some kind of commands to the robotic arm. 1 hand is to be imitated, while the other hand can be used to create gestures representing higher level instructions. The movements and gestures are described in Chapter 3. Whatever action of hand is captured by kinect, is translated into a instruction for robotic arm. Robotic arm doesn't have as many degrees of freedom as a hand, but it can reach any place a hand can reach because of its freedom in multiple directions.

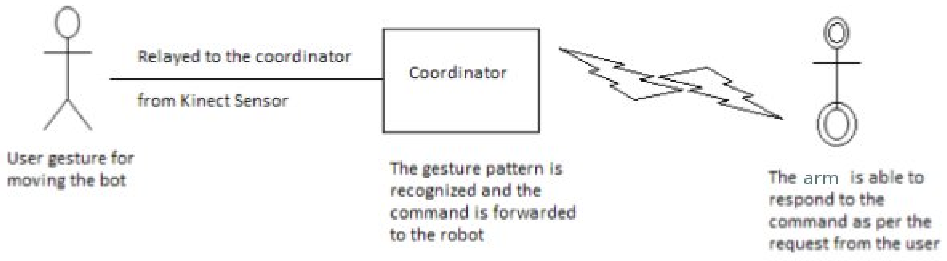

Relayed to the coordinator
from Kinect Sensor
Coordinator
User gesture for moving the bot
The gesture pattern is recognized and the command is forwarded to the robot
The arm is able to respond to the command as per the request from the user

Figure 2.1: Robotic Arm Control Process

2.3.2 Initial Step-By-Step Description

To run it smoothly, the user must be recognized by kinect and wireless connection between robotic arm and computer must be active and strong enough to transmit commands and camera pictures through it.

1. The user performs movement of right hand in front of kinect sensor.
2. The user can make use of gestures by left hand to start/pause robotic arm from imitating actions of user's right hand.
3. The gestures and movements are recognized by kinect and computer and the command is forwarded to robotic arm on wireless.
4. Robotic arm accesses the command and performs the required task.
5. Robotic arm captures the environment around it using the camera and sends it via wireless to computer.
6. Pictures received from robotic arm are displayed on computer.

2.4 User Characteristics

The user is expected to be able to perform the hand movements slowly taking good care that the kinect sensor is able to recognize the motion and ensure that most part of her arm is visible to the kinect sensor, so that it can be detected properly and analysed by the program on the coordinator which translated these motions into skeletal data. Main hand of user, which is to be imitated should not move around too fast, because kinect can capture around 10 frames per second, when arm is moving, and there is finite amount of lag between action and information. User should continuously monitor various interfacs presented to him, like the kinect sensor output, the skeletal data and the visual feed from the camera near the robotic arm.

2.5 General Constraints

1. The Kinect SDK is restricted to work on MS Windows 7 with DirectX SDK. Same will be the restrictions for the utility developed.

2. The Kinect Sensor Device requires a good processor and a dedicated VGA card. Same will be the requirements of the machine used as the coordinator.
3. The distance between the user and the kinect sensor must be 4 to 5 meters. The limit restriction is higher than that of kinect sensor, in order to prevent any interference from influencing the functioning of the Kinect Sensor.
4. Hand should not be moved around too fast, depending on the bottleneck speed of utility, most probably decided by wireless connection or kinect processing speed.
5. Robotic arm cannot perform all kind of hand movements, as it doesn't have exactly same freedom degrees as hand, although it should be able to full-fill the ultimate purpose of reaching some space coordinate.

2.6 Potential Applications

We intend to implement at least one of the following applications of the Kinect Controlled Robotic Arm (some of these may require implementation of additional features):

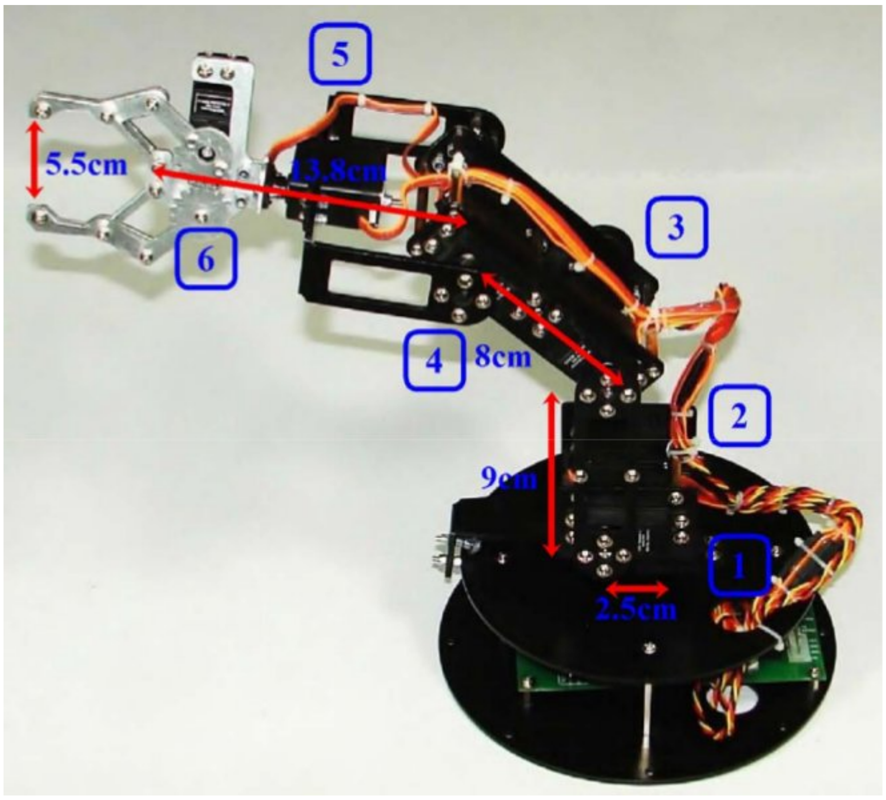
- Remote Bomb Diffusion Robot
- Remotely controlled Crane with NUI
- Writing/Painting using a Robotic Arm

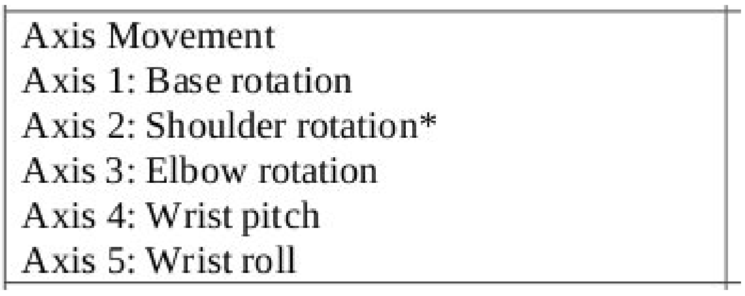
2.7 Requirement Subsets

- Hand movement Detection
- Hand gesture recognition
- kinect and computer connection
- Computer and ZigBee connection
- ZigBee and Robotic arm connection
- Interpretation and translation of commands
- Mounting camera on suitable position on Robot
- Map from hand movement to Robotic arm actions

3 Robotic Arm Control

The robotic arm is as shown in the figure. It has 6 axis of freedom corresponding to the 6 motors whose positions are shown. The movements corresponding to the 6 axis are presented in the table.





Axis	Movement
Axis 1:	Base rotation
Axis 2:	Shoulder rotation*
Axis 3:	Elbow rotation
Axis 4:	Wrist pitch
Axis 5:	Wrist roll

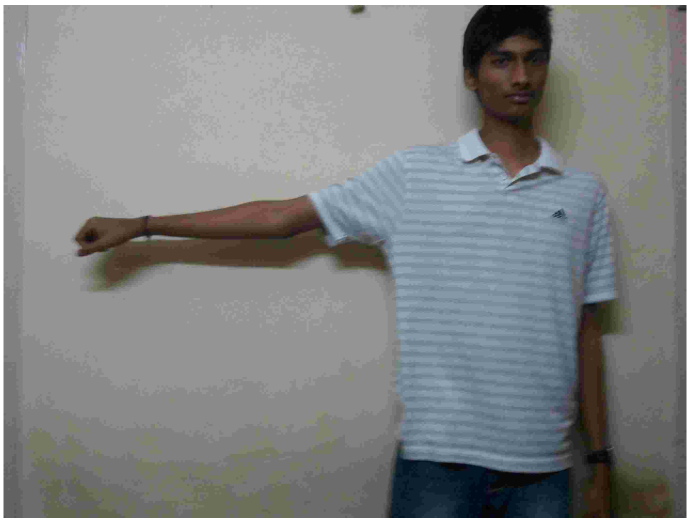
We have to map these movements to the various gestures of the hands of the user. Below we describe the various gestures and their corresponding actions.

Start and Stop

For this we can use the speech recognition capability of kinect. When the user says ”start”, the control of the robotic arm starts. When the user stays ”stop”, the control of the robotic arm ceases.

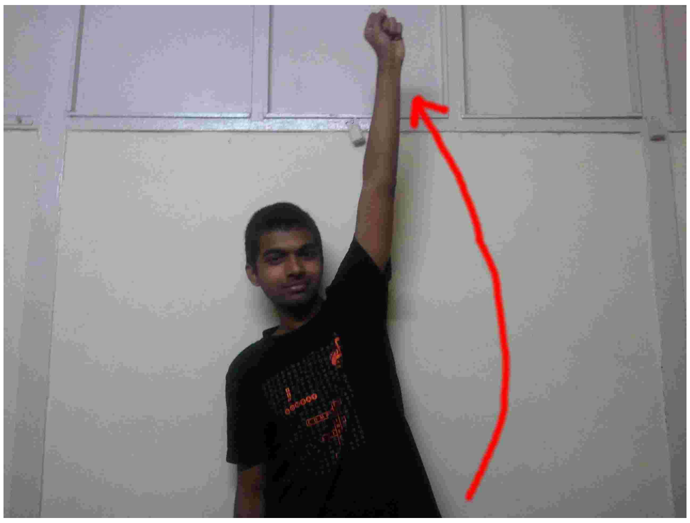
Zero Position

This corresponds to the position of the robot relative to which all movements are done. We can change the zero position of the robot using the hold gesture.

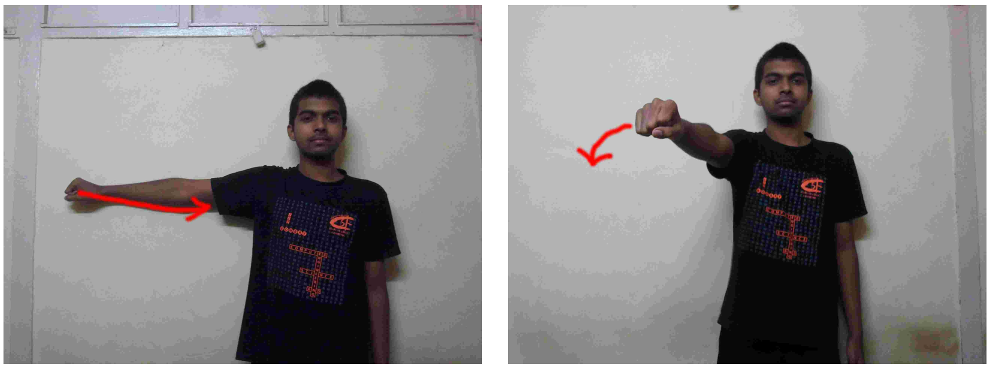


Hold

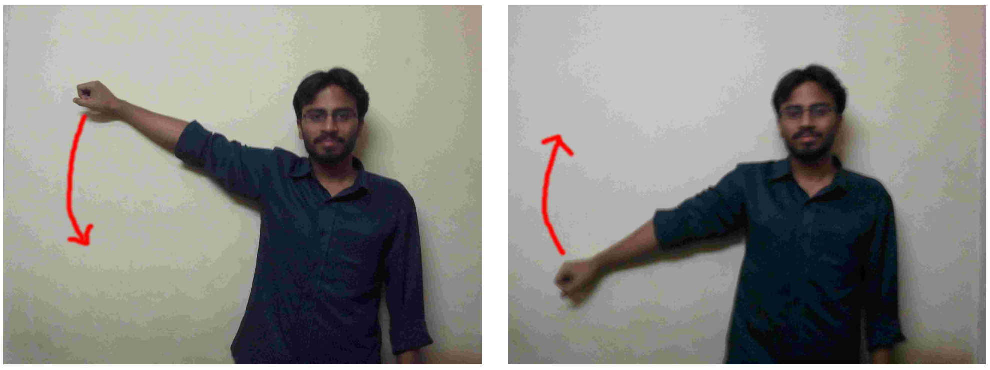
While we make the hold gesture, the control of the robotic arm stops temporarily. And this will make the current position of the robot to ’zero’. All further movements of the robotic arm will be relative to this ’zero’ position.



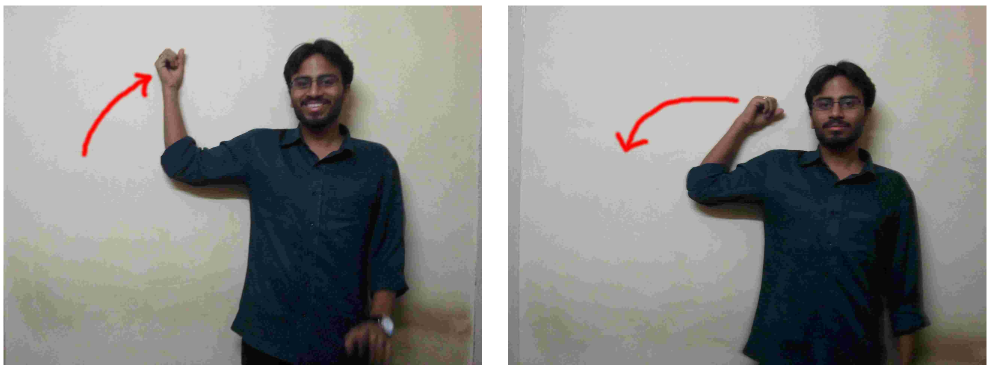
1.Base Rotation



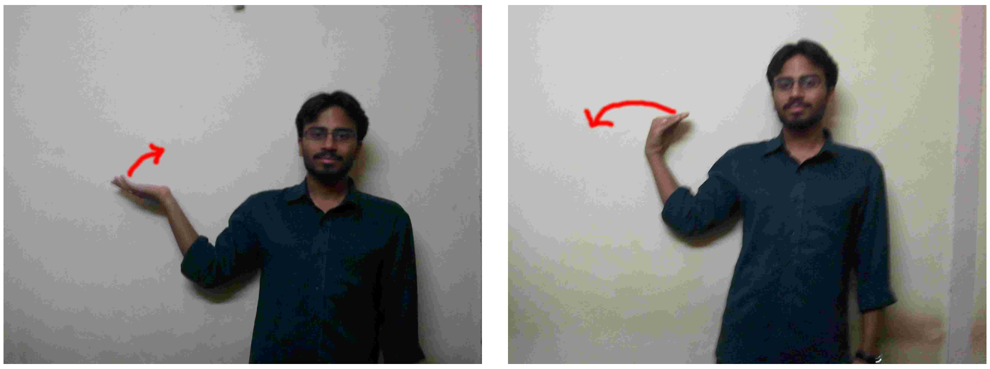
2.Shoulder Rotation



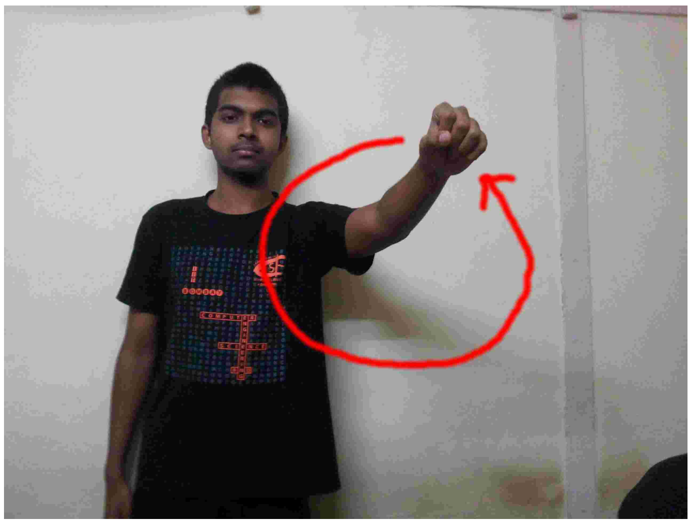
3.Elbow Rotation



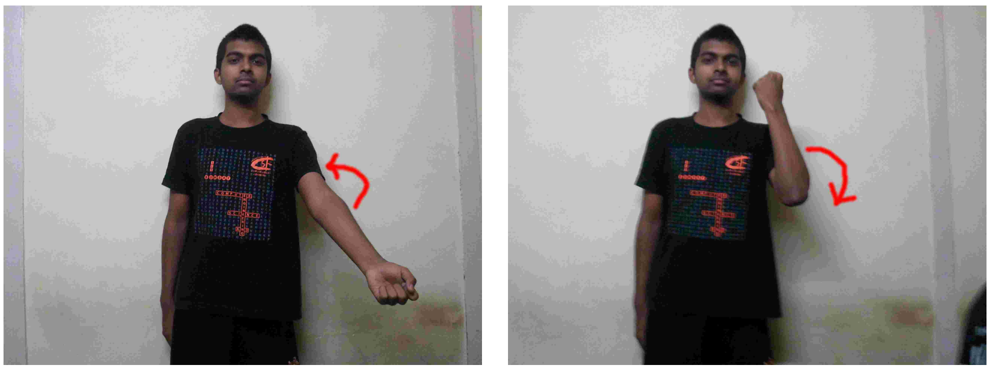
4.Wrist Pitch



5.Wrist Roll



6.Grip and Ungrip



4 Details

4.1 Functionality

Controlling a robotic arm with a traditional controller can be very difficult. It is very non-intuitive and increases the complexity of using the arm thus limiting its applications. With kinect the usability of the system is greatly enhanced because the natural movements of the human arm directly control the robotic arm which tries to imitate its movements. Thus controlling the arm becomes very intuitive and humans can readily use the robotic arm thus opening a plethora of applications.

This system is designed to allow a user to control the robotic arm via remote wireless connectivity. A camera is mounted with the arm so that the user can see the environment of the arm and the arm itself as live video which will help the user control the arm better. This will enable the user to deploy the arm in remote or hostile places thus further multiplying the scope of applications. The wireless network is configured to provide point to point connectivity between the coordinator and the arm ensuring no external interference on the communication channel.

Kinect will facilitate recognition of the movement of the user arm to be detected as skeletal data which in turn are converted to signals used to control the arm.

4.2 Supportability

4.2.1 Basis for future work

Because of its very general nature, this project can serve as the fundamental component of future projects exploring exciting applications opened up by this technology. Also we will try to abstract the most important parts of the system such that they are not specific to the robotic arm we are using or other specific details. So future projects can directly use this abstract high-level parts of our code without any modifications.

4.2.2 Distinguished Modules of the Project

- Kinect programming using Microsoft API for natural user interface for arm movement recognition
- Controlling the robotic arm using ZigBee communication
- Transmitting video data from remote camera to the coordinating computer

4.2.3 Documentation

Documentation in terms of embedded comments and other explicit write-ups will help others in future. We will try to provide high-level wrapping functions for all the important modules of our project and properly document them.

4.3 Design Constraints

The restrictions being implemented in the code are primarily due to those imposed of the system by the hardware being used. They are as follows:

- The human arm has an amazing degree of freedom and the robotic arm cannot exactly imitate. Thus we have to define appropriate actions for the arm for movements which directly do not correspond to human arm movements
- It will be very difficult to represent finger movements even if the robotic arm supports it. But in our project we ignore this detail except for the actions corresponding to gripping and ungripping
- The ZigBee Wireless Communication Module uses a Serial COM Ports. So, all the communication between the coordinator and the Robot will be via a serial port only
- The ER2 robotic arm has a payload limit of 50gms. This makes the arm mechanically too week for any realistic work.
- The ER2 robotic arm have a inbuilt Zigbee interface, so a Zigbee module needs to be attached to the arm for wireless control.
- Microsoft Kinect SDK and DirectX SDK require Microsoft Visual Studios to be used to allow proper access of their libraries. So, the use of such SDKs will require Visual Studios explicitly

4.4 Interfaces

4.4.1 User Interfaces

This is the part of the system that will be used to interact with the user i.e. recognizing their arm movements as well as displaying information for them to be used for controlling the robotic arm. They are as follows:

- Kinect interface view of the user's arm which includes the skeletal view, the camera view and the IR greyscale distance measurement view of the user's arm.
- A display showing the camera view of the robotic arm and the environment.
- The Robotic Arm GUI

4.4.2 Hardware Interfaces

This section states the essential physical equipment required to interact with the application designed. They are as follows:

- Kinect Sensor Module to detect the movements of the user arm
- Computer System with the following essentials are Kinect Sensor Module to run:
  - Intel Core2Duo 2.66GHz
  - Dedicated VGA card with DirectX Support

- A camera to be mounted with the robotic arm with the data bus connectivity to the ZigBee communication module.

4.4.3 Software Interfaces

This section states the requirements to develop the application which will use the data feed from the various interfaces and use them to control the arm or display information for the user. They are as follows:

- Kinect SDK which in turn requires the following:
  - Windows 7
  - DirectX SDK
- Visual Studios
- Atmel AVR Studios
- The Robotic Arm GUI

4.4.4 Communications Interfaces

This section states the requirements to create a communication between the robotic arm and the coordinator. They are as follows:

- 2 ZigBee Wireless Modules
- X-CTU ZigBee Configuration

5 Quality Control

Accuracy of Skeletal Data

The accuracy with which the arm can be controlled depends on the accuracy of the skeletal data obtained from Kinect. Better results can be obtained by considering the average over multiple frames.

Complexity of the Robotic Arm

The accuracy with which movements of the human arm can be reproduced on the robotic arm is limited by the complexity of the robotic arm. For example, the robotic arm may not have an arrangement for rotation of the wrist.

Complexity of the Algorithm

The skeletal structure of the human arm, and that of the robotic arm may be very different, and the accuracy of the reproduction of movements depends on the algorithm used to map movements about joints in the human arm to those in the robotic arm.

Calibration of Kinect for Gestures

The kinect uses an IR sensor and accuracy of the depth data is limited by the calibration of the sensor.

6 Risk Management

Protection Against Large/Fast Movements

Large or very fast movements of the human arm, if reproduced directly, can cause damage to the robotic arm. Care must be taken to limit the speed of movement and the range of angles covered by the robotic arm.

Protection Against Wear and Tear

The motors used at the joints may may be suffer damage if used continuously for extended periods of time. To prevent this, the motors must be automatically turned off when idle.

Protection Against Heavy Payload

The ER2 robotic arm has a payload limit of 50gms. Heavier payloads may cause damage to the motors. Care must be taken not to exceed this limit.

Problems with Calibration

Miscalibration of the depth sensor of Kinect can lead to incorrect and unexpected results. To avoid this, the calibration of the sensor must be checked regularly.

Problems with Gesture Recognition

A single pattern made by the user will never be an exact match to itself if tried repeatedly. So, the recognition program will have to be sufficiently lenient so as to allow some liberties to the user but not to misinterpret a general movement for a gesture. So, in order to confirm a gesture, a trial run will be needed prior to using the bot to confirm the movement pattern is recognizable as a gesture during the trial runs.