KINECT BASED REMOTE CONTROL OF HARVESTING BOT

CS 684: Embedded Systems Project Report

By

Ashok Nallagalva, 123059014

Jyoti Gajrani, 113053001

Pooja Mazumdar, 113050001

Satish Vemireddy, 123053001

under the guidance of

Prof. Kavi Arya

Prof. Krithi Ramamrithham

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

November 2012

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Statement

There has been a substantial increase in automated greenhouse culture in urban areas. One such example is a farmer's assistant harvesting bot that can be programmed to identify ripen fruits/vegetables using image processing algorithms and harvest them accordingly. But there have been flaws in identification of ripen fruits which lead to inefficiency in harvesting job. Since human is less error prone in identification of suitable fruits/vegetables that are ready for harvesting, a human controlled bot has been made which follows instructions given by a human. And all this would make sense if the human presence is not required in the greenhouse. This project facilitates remote access to such harvesting bot kept in greenhouse from anywhere in this world.

1.2 Overall Description

The IP camera kept at greenhouse gives a bird-eye view to the user at remote location through our GUI which is used by the person for controlling the bot. The kinect device placed at remote site (client) captures the gestures given by the user which is then sent to a system for generating bot commands. These commands are then transferred to a system kept in greenhouse (server) over Internet using socket programming which is then forwarded to the bot wirelessly over zigbee module. The bot (firebird V) and the gripper arm mounted on it, follow these commands and take appropriate actions for performing harvesting, which can be verified by the user by observing video sent from IP camera. Fig 1.1 provides an overview of the complete setup of this project.

This project is an extension of Team 16 : Gesture controlled Bot, Spring 2012 [6]. This documentation focuses mainly on the extensions done to the existing project. For detailed description of gestures, please refer [6]. Some gestures (particularly for arm movements) have been changed to make it more intuitive, details of which are provided in the project video.

1.3 Hardware Requirements

- Kinect sensor device
- Two computer systems (one acts as server and other as client)
- Firebird V ATMEGA2560

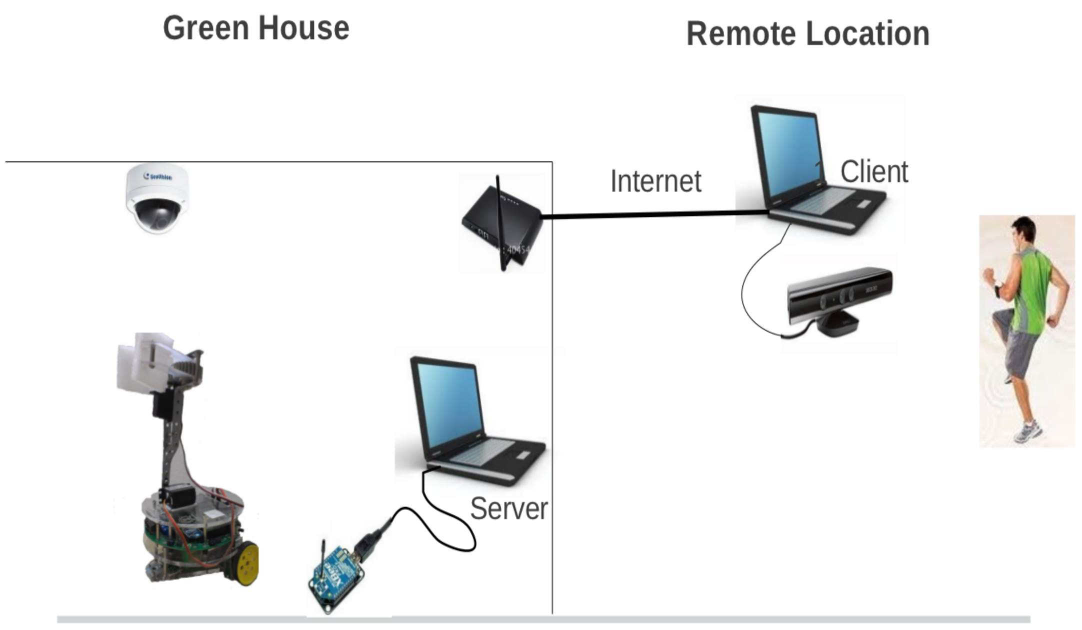


Figure 1.1: System overview

- X-bee for wireless communication between server system and bot
- IP camera for transmitting live video over internet to client

1.4 Software Requirements

- The Kinect SDK beta is used to work on MS Windows 7 with DirectX SDK
- The Kinect Sensor Device requires a good processor and a dedicated VGA card.
- X-CTU for configuring X-bee
- IPCam Admin Utility for configuring IP camera
- Microsoft Visual Studio 2010 for development of client and server applications in C#
- .NET framework 4.0
- AVR Studio for writing firebird code
- AVR bootloader for burning code to bot

1.5 Extension done in firebird

A gripper arm is attached to the firebird for harvesting fruits/vegetables. Two servo motors have been used for this. One servo motor controls the vertical movement of the gripper arm and the other controls the opening and closing of the arms. Thermocol is attached to the arm to provide better grip (details in hardware manual).

Chapter 2

System Design

2.1 Product perspective

Providing a time-saving and cost-saving way of managing multiple greenhouses from remote location is the prime motivation behind this project. This product is useful to control the firebird without any knowledge of hardware (bot). Gestures used for controlling the bot are very intuitive and can be learned easily by any layman.

2.2 User Flowchart

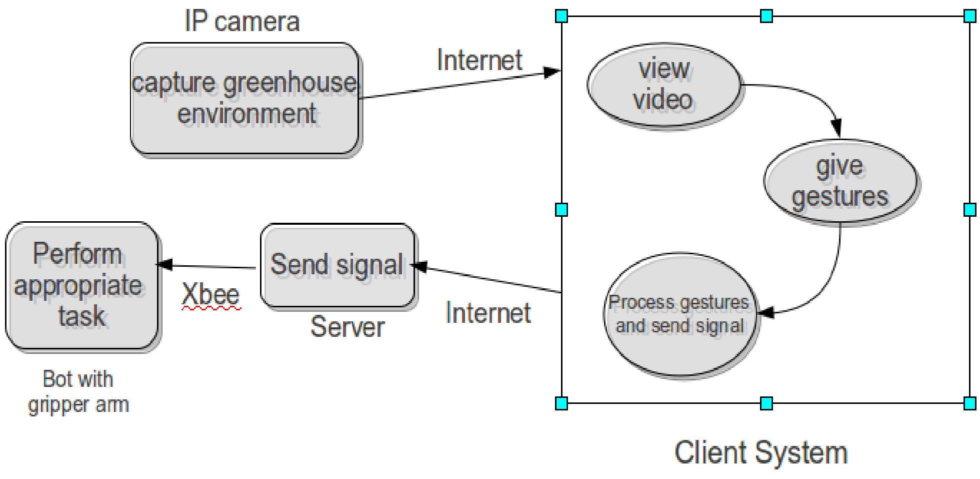


Figure 2.1: Flow chart of the complete setup

2.3 User characteristics

User is expected to have basic knowledge of Kinect and should know the appropriate predefined gestures to be able to control the bot and the arm.

2.4 Statechart

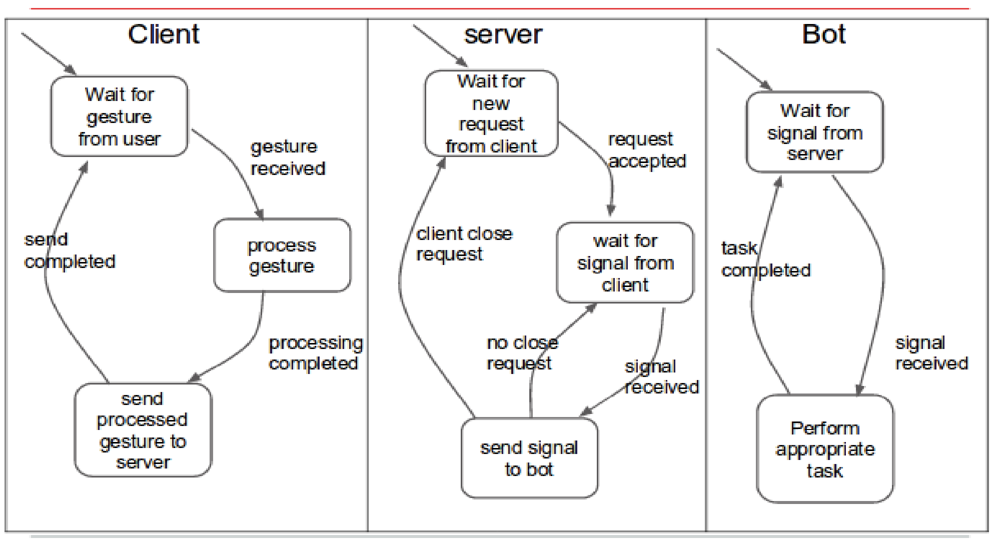


Figure 2.2: Statechart describing the working of the project

2.5 Constraints and assumptions

1. Very fast gestures cant be detected correctly. This is because the range of data transmitted by X-bee is 0-128. so we have scaled down values to bring it in this range which may lead to some inaccuracy.
2. No disturbance in the background of person giving the gesture is assumed
3. Currently it accepts gestures from only one user at a time

2.6 Hardware and Software setup

Refer respective manuals

Chapter 3

Implementation

3.1 Execution Instructions

1. Do all the connections and configurations of ZigBee, IP Camera and Gripper arm as mentioned in hardware manual
2. Refer software manual to install required softwares to run this project
3. Burn the firebird code (firebird/Serial_Communication_ZigBee_wireless/Serial_Communication_ZigBee_wireless/default/Serial_Communication_ZigBee_wireless.hex) on firebird V
4. On server system, open "ConsoleApplication1.sln" (in folder server_project/) file in visual studio and run the application
5. On client system, open "SkeletalViewer.sln" (in folder client_project/) file in visual studio and run the application
6. Provide proper gestures at client side (as demonstrated in the project video)

3.2 Code Description

3.2.1 Code Files

The various code files along with the purpose they serve and on which component of the project they are executed, is described in Table 3.1.

Filename	**Purpose**	**Executes on**
gesture_firebird.c	bot code	Bot
MainWindow.xaml.cs	processing gestures and transmitting to server	client
Program.cs	transmit signal to bot	server

Table 3.1: Code files

3.2.2 Project Files

The various project files and their usage, is described in Table 3.2.

Folder Name	Usage
client_project.zip	client application
server_project.zip	server application
firebird.zip	bot code

Table 3.2: Project files

3.2.3 Client Code

The client side serves 2 main purposes: displaying video of greenhouse received from IP camera and processing user gestures sent through kinect device. These 2 modules have been integrated in a single GUI. The upper part of the interface displays the greenhouse video and lower part displays the kinect skeleton and user video captured by kinect camera. Based on the given gesture, kinect device sends angles corresponding to all the hand joints to the client software, based on which the client code generates the corresponding command and sends it to the server using TCP/IP socket programming. It checks for various angles repeatedly and then after scaling (if needed), writes on the socket after every 10 frames.

3.2.4 Firebird Code

Detection of signals sent by X-bee wireless module is implemented in ISR for UDR0. Values of front IR and sharp sensor are checked first to avoid any collision. This extension was done to handle delay in receiving **stop** command at appropriate time. This delay can occur due to congestion in the network (between client and server) or can be caused due to delay in receiving greenhouse video. We have observed 1 second delay in video feed received from IP camera (mentioned in Delay analysis section). The value of received signal is stored in variable **data**. As all the gestures are divided in 3 modes, first of all mode is set based on the value of **data** and then either a specific value or a range of values is checked and the motion of bot/arm is decided. For the motions of gripper arm, servo motor functions are called with appropriate values.

3.2.5 Server Code

TCPListner is initialized which listens on port 8001 and waits for clients to connect to it on the server's IP address. Once the connection is accepted, it opens a socket and continuously receives data on the opened socket. When the client wants to close the connection, it sends a string **End** to the server. As soon as the server receives this string, the socket connection is closed and server waits for a new connection.

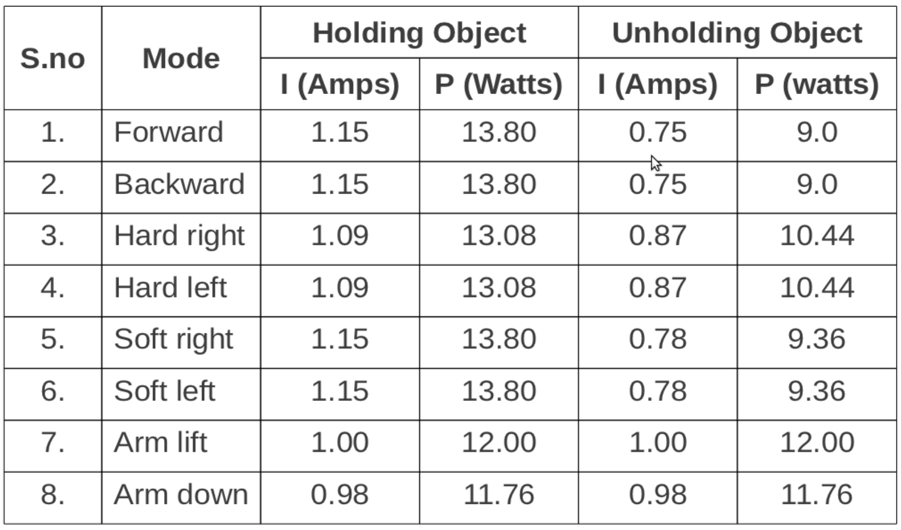
Chapter 4

Performance Metrics

4.1 Delay Analysis

- The IP camera have delay of approximately 1 second in transmission of video. This leads to incorrect guess of exact location of bot, which may lead to wrong gesture transmission.
- Network delay is around 51.2 microseconds
- Kinect delay is in the order of milliseconds

4.2 Power Analysis



S.no	Mode	Holding Object		Unholding Object	
		I (Amps)	P (Watts)	I (Amps)	P (watts)
1.	Forward	1.15	13.80	0.75	9.0
2.	Backward	1.15	13.80	0.75	9.0
3.	Hard right	1.09	13.08	0.87	10.44
4.	Hard left	1.09	13.08	0.87	10.44
5.	Soft right	1.15	13.80	0.78	9.36
6.	Soft left	1.15	13.80	0.78	9.36
7.	Arm lift	1.00	12.00	1.00	12.00
8.	Arm down	0.98	11.76	0.98	11.76

Figure 4.1: Power analysis chart

When the bot is fully charged, it can provide 2.1 Amperes for 2 hrs. Based on this and observing from Fig 4.1, we provide the time estimation for which bot can run on battery power:

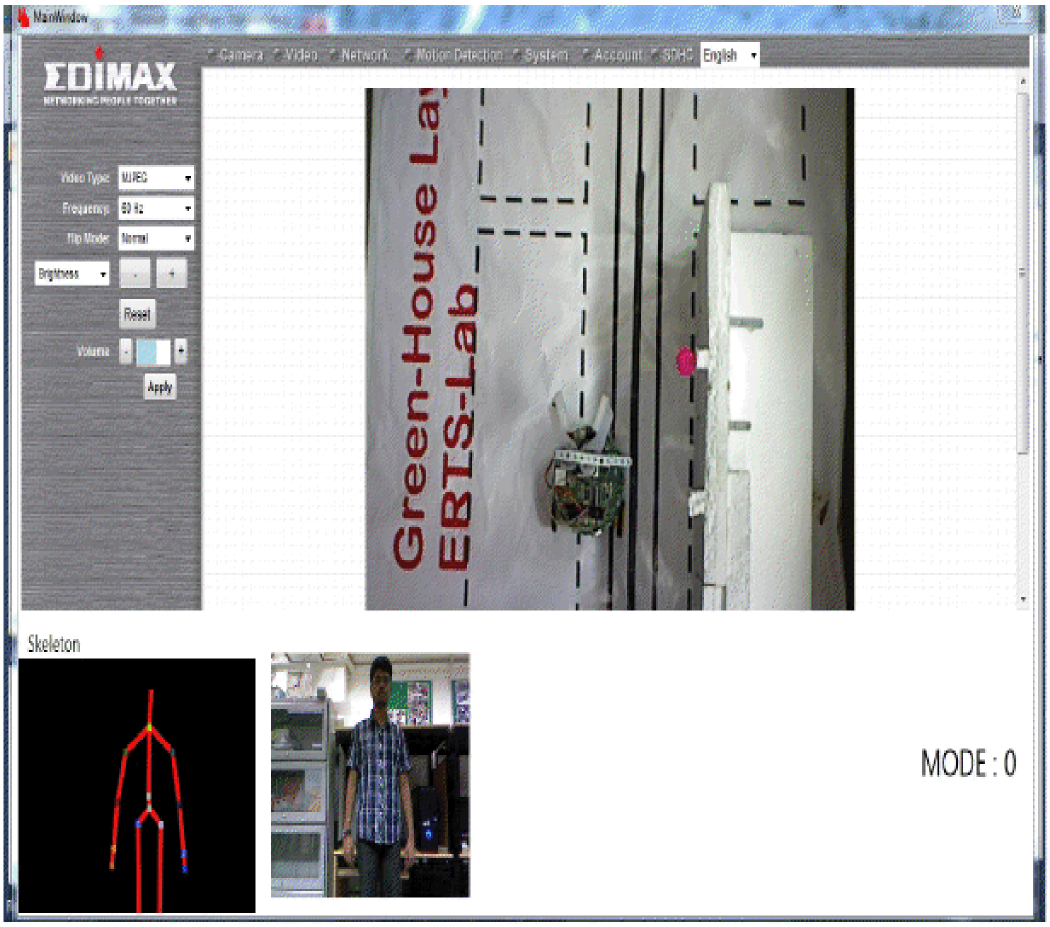
- Worst case (assuming the case in which bot takes maximum current) : 3 hrs 39 mins
- Best case (assuming the case in which bot takes minimum current) : 5 hrs 41 mins
- In Harvesting (assuming the case in which bot takes average current) : 4hr 22 mins

Average power consumption of bot in harvesting one fruit is : **11.63 W**

Chapter 5

Interfaces

5.1 Client Interface

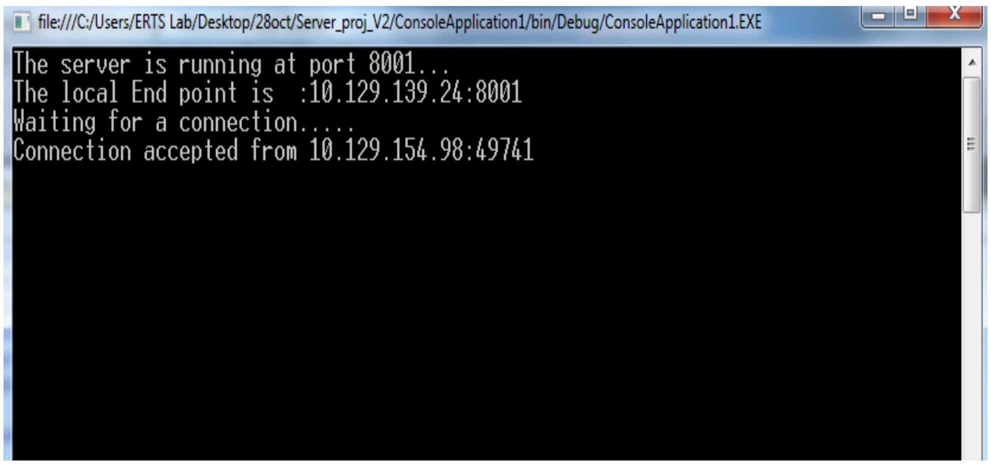


Video Type: MJPEG
Frequency: 60 Hz
Flip Mode: Normal

MODE : 0

Figure 5.1: GUI used at client side

5.2 Server Interface


file:///C:/Users/ERTS Lab/Desktop/28oct/Server_proj_V2/ConsoleApplication1/bin/Debug/ConsoleApplication1.EXE

```
The server is running at port 8001...
The local End point is  :10.129.139.24:8001
Waiting for a connection.....
Connection accepted from 10.129.154.98:49741
```

Figure 5.2: Server application

5.3 Bot with gripper arm

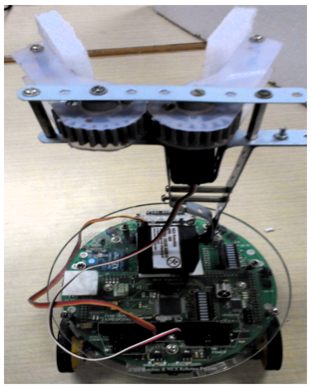


Figure 5.3: Firebird with gripper arm

Chapter 6

Testing

6.1 Unit testing

- Verification of gesture recognition by printing their name and angle values
- X-bee verification by data transmission using 2 terminals on 2 different ports
- IP camera testing in browser
- Independent testing of communication between client and server modules
- Deciding delay value for synchronization between two continuous values

6.2 Integrated testing

It was done by placing the bot on the provided arena and controlling it by giving proper gestures such that the bot moves on the aisle and stops near the trough (thermocol wall) where fruits (balls) are placed. The bot is then instructed to reach to a fruit (ball) and pluck it. While reaching to the fruit, collision control was also tested as the bot stops automatically detecting the trough.

Chapter 7

Challenges

7.1 Problems that were handled

1. Kinect gestures should not overlap with each other. Composition of kinect gestures had to be done carefully. Adding new gestures, that do not overlap with existing ones, was difficult.

2. Fluctuations in data received from kinect between every two frames and sending this continuous data to the bot may cause jerks and damage to motors. Therefore, data after every 10 frames were sent.

3. Due to delay in receiving video of greenhouse at the client, exact position of bot was difficult to estimate at the time of sending gestures. Collision control is implemented to prevent the bot from hitting objects if **stop** command is not received in time.

4. Mismatch between data sending speed of system (high) and data receiving speed of firebird (low) causes problem. To bridge this gap, delay was introduced between 2 continuous values.

5. Transmission of commands from client to server system over Internet is erroneous. Since we must avoid data loss, TCP/IP socket programming was used for reliable transmission.

6. Hardware problems, like faulty position encoders, were handled by switching to a different bot with properly working components.

7.2 Problems that were not handled with solutions

1. The aisle width provided in arena for testing was very less. Turning bot in such small passage was very difficult. A solution to handle this problem is to add more degree of freedom to the gripper arm using one more motor such that the arm is able to move left and right. This eliminates the need of bot rotation.

2. Speed mismatch between left and right wheels of Firebird does not let the bot move straight in **forward** command if the bot code uses pin configuration for motion. The solution is to provide motion of the bot using PWM method. Since this fault was very unexpected and detected at the time of testing, changing the bot code at that time was risky. Also, switching to another bot didn't help because almost all bots have this problem.

3. Errors in estimating the exact position of fruits by looking in the video feed was detected while testing. For example, top view does not help in estimating the depth at which ball (fruit) is fixed on the thermocol wall. A solution to this problem is to use more than one IP Camera to capture different angles of the greenhouse e.g. one top view and one side view can be used to detect exact position of the ball mentioned in the above example.

Chapter 8

Future scope

- A robotic arm, in place of a gripper arm, can be used to increase degree of freedom.
- A local basket can be attached to the bot to speed up harvesting process.
- Speech feature of kinect can be used.
- More than one IP camera can be used to capture different angles of greenhouse. Need of multiple IP Cameras is clearly explained in Section 7.2.
- Arm rotation with one additional motor to reduce the space requirement for left and right movement of the bot. This requirement is also explained in Section 7.2.

Chapter 9

Road Map

The road map that led to project completion is mentioned in Table 9.1.

S.No.	**Milestone**	**Completion Date**
1	Project finalization	25-09-12
2	SRS submission	2-10-12
3	Identifying modules	5-10-12
4	Configuration of X-bee	7-10-12
5	Setup of IP Camera	10-10-12
6	Arm design	15-10-12
7	Gesture composition	20-10-12
8	Client application development in C# for processing gestures	15-11-12
9	Server application development in C#	20-11-12
10	Communication using Socket programming	23-11-12
11	Project Integration	28-11-12

Table 9.1: Road map of the project

Chapter 10

Conclusion

Apart from harvesting, this project can be used for surveillance of the greenhouse since the GUI at client side receives continuous video of greenhouse. As the bot follows instuctions given by human, it can be used for cleaning greenhouses and with some modifications in the arm, it can be further used for seed sowing and weeding purpose as well.

All the requirements that were mentioned in project SRS during initial stage of the project, has been completed with addition of one more functionality (Collision control - refer section 3.2.4). The project can be reused in future for adding more such automated functionalities to improve harvesting efficiency. This report provides a detailed description of the project using various diagrams, flow charts, statecharts and snapshots. We have measured performance metrics like delay and power consumption (described in chapter 4). Some degree of automation can be used to lower the power consumption. The challenges we faced during the development of this project and how we have overcome these problems are thoroughly described in this report. Some challenges which we faced during testing phase have not been handled due to shortage of time but we were able to provide appropriate solutions to those problems which are clearly explained in this report (refer chapter 7).

Chapter 11

Individual Roles and Contribution

Configuration of Zigbee module, Implementation of new gestures, Addition of collision control functionality in bot code and Extension in existing project's code to include socket programming is done by Pooja and Jyoti. Configuration of IP Camera, Gripper arm construction, Development of a web browser to display greenhouse video stream received from IP Camera, Designing client side GUI and Gesture composition is done by Ashok and Satish.

Bibliography

[1] http://www.i-programmer.info/programming/hardware/2822-getting-started-with-microsoft-kinect-sdk-depth-and-video-space-.html

[2] http://channel9.msdn.com/series/KinectSDKQuickstarts/

[3] Kinect for Windows SDK v1 : Resources and Documentation, Microsoft Corporation http://www.microsoft.com/en-us/kinectforwindows/develop/resources.aspx

[4] AVR AtMega2560 Programming Manual, Atmel, May 2011, http://www.atmel.com/Images/doc2549.pdf

[5] http://www.youtube.com/watch?feature=player_embedded&v=VzgUtZyiHhE#at=75

[6] http://www.e-yantra.org/home/projects-wiki/item/197-gesture-controlled-robot