**CS 308 – 2011 Project**

**GROUP 15**

**Project: SLAVE BOT**

**Varun Reddy 09005066**

**Tarun Gujjula 09005062**

**Ravi vooda 09005061**

**Rajesh Kamineni 09005062**

Table of Contents

[INTRODUCTION 3](#_Toc308991583)

[PROBLEM STATEMENT 3](#_Toc308991584)

[REQUIREMENTS 4](#_Toc308991585)

[Software Requirements: 4](#_Toc308991586)

[Hardware Requirements: 5](#_Toc308991590)

[IMPLEMENTATION 6](#_Toc308991598)

[System architecture 6](#_Toc308991599)

[Code folder 6](#_Toc308991600)

[Gesture Recognition in CSharp 7](#_Toc308991601)

[Communication with spark implemented in CSharp 7](#_Toc308991602)

[Camera View in CSharp 7](#_Toc308991603)

[Logic in CSharp 7](#_Toc308991603)

[Controlling Firebird robot using Gestures 9](#_Toc308991603)

[TESTING STRATEGY AND DATA 11](#_Toc308991604)

[Test Data 11](#_Toc308991605)

[DISCUSSION OF SYSTEM 11](#_Toc308991606)

[Observations from Testing 11](#_Toc308991607)

[General discussion 11](#_Toc308991608)

[FUTURE WORK 12](#_Toc308991609)

[CONCLUSION 12](#_Toc308991610)

# INTRODUCTION

The purpose of this document is to present a detailed description of the Gesture Controlled Slave Robot. 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 and how the system will react to external stimuli. This document is intended for the developers of the system.

The Gesture Controlled Robot has two active systems and one coordinating system. The communication between the coordinator and the robot is on a dedicated wireless frequency channel. The User is able to view the problem if any are reported by the bot on the coordinator.

The robot (Firebird V) is controlled remotely using the ZigBee wireless interface. The user is providing the input by movements representing gestures that are captured by the Kinect sensor device. The gestures captured by the Kinect sensor are translated to a particular command using the Kinect API and send signals to the robot. Kind of gestures we are implemented are, on pointing our finger to a object kinect will send the location of the object to robot. Bot will go to the directed location backtrack to its original position.

# PROBLEM STATEMENT

The objective of this project is to program the Firebird V robot, so that it can move to the position pointed by user relative to itself and backtrack itself to the starting position.

# REQUIREMENTS

Software Requirements:

The software is a windows based application. The software requires .Net4 Framework, Zigbee FT232R USB UART driver, Kinect Software and Camera Drivers.

### .Net Framework

This is required for further development of the software. We assume that the further development will be done only in MICROSOFT VISUAL STUDIO’s environment and .Net Framework is required for this.

### Zigbee FT232R USB UART Driver:

The medium of communication with the bot is through a wireless modem called ZigBee. Hence the drivers for this hardware are needed. Scilab and SIVP toolbox

### Kinect SDK :

The project requires input from a Kinect or Xbox 360 device. The software which communicates with the Kinect or Xbox 360 device as well as gives us required inputs is called Kinect Software. Hence Kinect software is needed. The driver of Kinect comes along with the Kinect Software.

### Camera Devices drivers:

The project has a camera system. Hence all the drivers required for the camera normal functionality are required for this software.

## Hardware Requirements:



### Spark-V

Spark V robot is based on ATMEGA16A microcontroller. Robot comes with rechargeable 7.2V 600mA NiMH Battery and onboard intelligent battery charger. Robot has onboard socket for Zigbee wireless module for multi robot and robot to PC communication.

### 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.

### XB24-ACI-001.jpgTwo Zigbee Modules

Zigbee is a wireless technology which address the unique needs of low-cost, low-power wireless M2M networks. One zigbee is placed on the bot whereas other is connected to pc through a USB cable.

### 2-4g-wireless-camera-803t.jpgWireless Camera, Receiver and TV Tuner

Wireless camera transmits the analog signals which is received by the receiver and TV tuner sends television signals to PC. This video can be received by setting this as video object source in C#.

# 

# IMPLEMENTATION

## System architecture

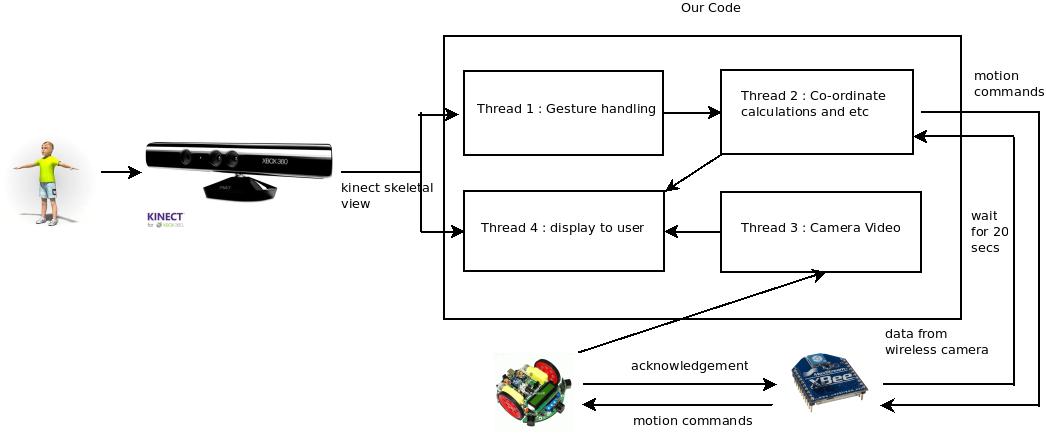


Figure 5 : System architecture

## Code folder

gestrecogn – Visual Studio Project Folder

This folder contains the Visual Studio (2010) project for gesture recognition module.

botcode – Scilab program

The code running on the bot in our project is located in this folder.

## Gesture recognition in CSharp

Gesture recognition is developed using Visual Studio 2010 using CSharp based Kinect Module. The CSharp project for the Gesture processing module is in the folder **GESTRECOGN.** This module outputs various frames like Color Frame, Skeleton Frame, Depth Frame etc. i.e. Photo related frames, which will be used in this project for simulating a realistic 3d world.

## Communication with spark implemented in CSharp

Serial Communication with spark is done by the CSharp with Ports interface available in System.IO. Since Zigbee communicates in ASCII language, we scaled down the values within ASCII values. (Distance is scaled down to 2, Angle to 3) An ACKNOWLEDGEMENT is expected in return with a timeout of 20 sec.

Pseudo code for serial communication through a USB port:

|  |
| --- |
| open the serial communication interface with port no = 6  Write to PORT; // Parameters : ASCII chars to be sent  Wait for ACK; |

## Camera View in CSharp

The view of bot is depicted by the camera on the bot, which is transmitted over a wireless medium to the application. These camera’s frames are rendered and displayed on the screen.

Pseudo code for Camera Display:

|  |
| --- |
| Set the properties of the Camera Display like frame rate etc.;  Wait For a Frame;  Display the Frame; |

## Logic in CSharp

A lock mechanism to lock a gesture is implemented in this project. From the locked gestures the co-ordinates of the point to which the bot has to move are computed and from these co-ordinates the angle of rotation and distance to be moved are computed and these values are stored in a queue.

These computed values are de-queued sequentially and are sent to bot to perform the required actions. When the queue is emptied backtracking sequence is initialized.

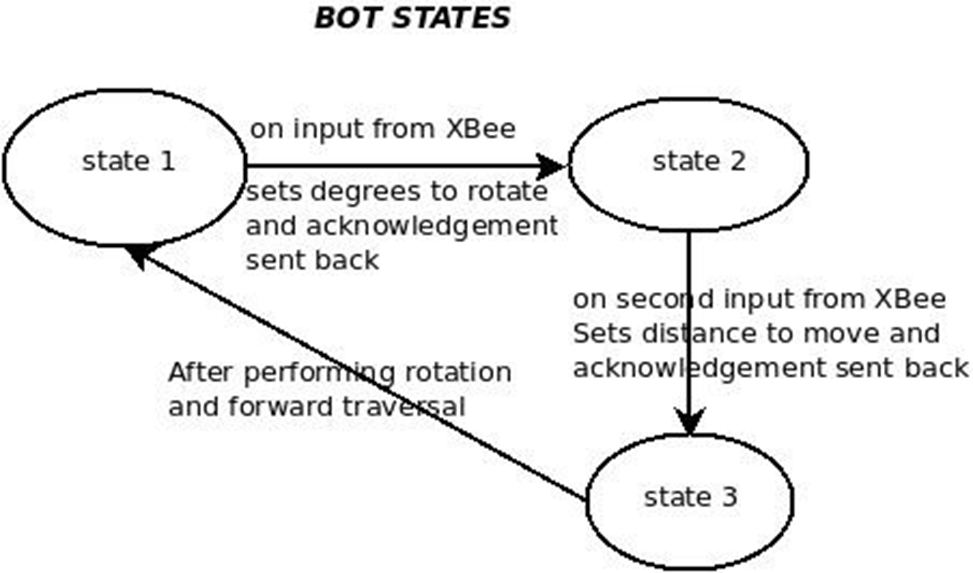


Figure 6: State diagram for bot code

## Controlling Firebird robot using Gestures

The gestures implemented by us are given below





# TESTING STRATEGY AND DATA

## Test Data

1. Identification of Various Gestures

Correctness of Gesture

1. Action of the bot

Observing the Bot’s movement

1. Zigbee testing

Acknowledgements

# DISCUSSION OF SYSTEM

## Observations from Testing

|  |  |  |
| --- | --- | --- |
| **Test** | **Observation** | **Reason** |
| Identification of various gesture | Various gestures were presented to the system and it worked very well | - |
| Action of the bot | The movement of the bot was very close to the expected movement | Frictional impurities are not considered. Bot is not in a perfect shape. |
| Zigbee testing | Acknowledgements are observed for loss. No loss detected. We expect 95 % efficiency in this module. | - |

## General discussion

1. The gestures implemented can be further improvised for a better recognition.
2. ‘Stop and Wait’ protocol implemented in the ACKNOWLEDGEMENT module.

Protocol makes sure transmitted signal is received by sending repeated signals till acknowledged.

1. The image processing part is completely implemented by Kinect Software.
2. Obstacle Avoidance has not been implemented and can be a useful extension of this work.

# FUTURE WORK

1. The mobilization part is developed in a thread environment, making it possible for any other projects to be confident regarding the mobilization part.
2. Obstacle Avoidance is a very important add-on to the project.
3. The algorithms can be further optimized and threading system can be further developed for a better scalable application.

# CONCLUSION

Our project is developed from scratch. Succinctly, through our project we were able to maneuver the bot with good precision, improve the computational time by effective use of threading, and provide a pragmatic execution of commands by stop and wait protocol.