SHIVAJI UNIVERSITY

Snake Catcher Robot

by

Author Name

Department of Computer Science and Engineering
Sharad Institute of Technology College of Engineering,Yadrav-Ichalkaranji

April 2015

SHIVAJI UNIVERSITY

Abstract

Department of Computer Science and Engineering
Sharad Institute of Technology College of Engineering,Yadrav-Ichalkaranji

Bachelor Of Engineering

In this project, it is aimed to control a robot using wireless technology. The robot is able to catch and release the snake and to move correctly. We give a direction to robot manually. When the robot gets the direction command, it moves according to command also robot gets catch command it catches the snake correctly.

This is a robot whose motions can be controlled by the user by giving specific commands, for wireless communication we are using Xbee Module. This is capable of identifying the 6 commands Run, Stop, Left, Right and Back issued by a particular user. What we are aiming at is to control the robot using following Commands. Robot which can do these basic tasks: Move Forward, Move Backward, Move Left, Move Right, Stop. Robot which can do these main tasks: Catch, Release. The objective of this project is to a develop Robot which is capable to catch snake by taking wireless commands.

.

Acknowledgements

We Will Fill it.

Mr. Vinay Shamrao Kumbhar

Mr.Sushant Mahavir Patil

Mr.Suhas Popat Zanjad

Mr.Sharad Chandrakant Asabe

Contents

List of Figures

Chapter 1

Introduction

1.1 Introduction about project

Robot: It is an electromechanical device which is capable of reacting in some way to its environment, and take autonomous decisions or actions in order to achieve a specific task[1]. This means that a toaster, a lamp, or a car would not be considered as robots since they have way of perceiving their environment. On the other hand, a vacuum cleaner that can navigate around a room, or a solar panel that seeks the sun, can be considered as a robotic system. It is also important to note that the robots featured in Robot Wars for instance or any solely remote controlled device would not fall under this definition and would be closer to a more complex remote controlled car. Although this definition is quite general, it might need to evolve in the future in order to keep up with the latest advancement in the field. In order to get a sense of how robotics is rapidly growing, we suggest you take a look at the Robot Shop History of Robotics[2]. What does an easy to make robot look like? May seem like a dumb question, but we promise a good look at other robots will really help us. Browse the web for robot galleries. Remember, only bother looking at the really simple robots, DO NOT get imaginative or creative on your first robot. You're only asking for trouble if you want to make something like a biped with camera vision and an arm to get wet a beer. What we want is a dumb little 2 wheeled thing that just doesn't commit suicide by driving off your table. And for a beginner, that can be extremely hard in it[3]. Radio control (often abbreviated to R/C or simply RC) is the use of radio signals to remotely control a device. The term is used frequently to refer to the control of model vehicles from a hand-held radio transmitter. Industrial, military, and scientific research organizations make [traffic] use of radio-controlled vehicles as well[4]. A remote control vehicle is defined as any mobile device that is controlled by a means that does not restrict its motion with an origin

external to the device. This is often a radio control device, cable between control and vehicle, or an infrared controller. A remote control vehicle (Also called as RCV) differs from a robot in that the RCV is always controlled by a human and takes no positive action autonomously. One of the key technologies which underpin this field is that of remote vehicle control. It is vital that a vehicle should be capable of proceeding accurately to a target area; maneuvering within that area to fulfill its mission and returning equally accurately and safely to base. Recently, Sony Ericsson released a remote control car that could be controlled by any Bluetooth cell phone. Radio is the most popular because it does not require the vehicle to be limited by the length of the cable or in a direct line of sight with the controller (as with the infrared set-up)[5]. Bluetooth is still too expensive and short range to be commercially viable. Snake catcher robot can be used to traverse many different and difficult places, such as holes, tunnels and gaps,. Snake catcher robot usually have more DOF than is necessary for a given task, thereby providing the robot configuration with a certain degree of redundancy. Numerous Snake catcher robot designs have been developed, however, various robot designs differ greatly in physical configurations and purpose. Project of Snake catcher robot designs into five categories: (1) robots with passive wheels, (2) robots with active wheels, (3) robots with active Arm, (4) robots based on undulation using vertical waves and (5) robots based on undulation using linear expansion. In each of these categories, several robot designs were presented and selected robot designs were discussed in detail[6]. Based on the plot of robot data the following general observations can be made. First, it can be assumed from the velocity-length plot that a faster robot may have a longer length in comparison to another Snake catcher robot. This assumption may be supported by the idea that a longer robot may have additional propulsive force due to additional active joints or modules. Second, at present, Snake catcher robot executing rectilinear-based locomotion are typically slower than robots which use passive and active wheels. This assumption is based on a comparison between the Category IV robots and the remaining wheeled robots[7]. It should be noted that there are not enough robots in each category to compare the categories; however, there are enough to compare pure undulation-based robots and wheel-based robot designs. Finally, it was determined that accurate comparisons between the various Snake catcher designs cannot be achieved using a single performance metric. An accurate comparison would require that the designer use several metrics to include the robots performance, configuration, and intended operating environment. Although snake-inspired robot designs have demonstrated general functionality and a number of useful gaits, the current designs still have not been placed into widespread use. Even with better understanding of the current proven designs and their useful features, there are still major design challenges which future designers must resolve to increase the practicability of Snake catcher robot[8].

Chapter 2

Literature Review

2.1 Literature review of project

Worldwide investment in industrial robots up 19 percent in 2003. In first half of 2004, orders for robots were up another 18 percent to the highest level ever recorded. Worldwide growth in the period 2004-2007 forecast at an average annual rate of about 7 percent. Over 600,000 household robots in use - several millions in the next few years. UNECE issues its 2004 World Robotics survey. From the above press release we can easily realize that household (service) robots getting popular. This gives the researcher more interest to work with service robots to make it more user friendly to the social context. Wireless technology gives the researcher the opportunity to no need of add Natural language (NL) communication with robot in natural and even way in the social context. So the promise of robot that behaves more similar to humans is starting to become a reality. Brooks research is also an example of developing humanoid robot and raised some research issues. Form these issues; one of the important issues is to develop machine that have human-like perception[3].

There are a few things to remember when dealing with snakes: Snakes are not toys. Only catch a snake for qualified research purposes, or if you absolutely have to. Remember, its stressful for the animal to be captured. Start small. Don't try catching a 2-metre cobra on your first day out, you'll come off second best. Start with practicing the wrist movements and techniques on a rubber snake or a thick piece of rope. The rope works well because if it's as thick as a snake, it reacts a little bit like one in terms of flexibility. Once you're confident in that, move onto a harmless species, and practice on it. Preferably use a specimen bred in captivity as opposed to a wild snake[9]. Captive-breeds are more used to human interaction and you'll therefore cause it less stress. Know your species. Pretty basic, but important. Know what snakes

you're likely to encounter around your area, some are more aggressive than others etc etc etc. Focus on the snake. Don't answer your phone with a snake in the hand, again, you'll come off second best and this is probably the most important thing to remember when dealing with venomous snakes: COMPLACENCY KILLS. Just because you've caught a hundred snakes successfully doesn't change anything. Finally, this is obviously just a guide - I recommend attending a handling course or something similar where a professional can teach you face to face about dealing with snakes. Catching a massive, venomous snake is dangerous business. But what if there was a safe way to capture the creature from a safe distance? Robots are often used for jobs that are too dangerous for humans, such as defusing bombs, capping oil wells deep down in the ocean, even exploring other planets. But can a robot take over the potentially deadly job of catching snakes? The Snake Catcher Robot will have to operate in the darkness and function in tiny, confined spaces. Itll certainly need a camera. Perhaps some lights, too. It should be able to investigate both high and low places. And finally, the Snake Catcher Robot operator must have full control of the robot while remaining out of harms way[10]. So, whats it take to transport, manipulate and operate the Snake Catcher Robot? What will happen when its finally put to the test? In this project, our aimed is to control a robot using remote control. The robot is able to understand remote controlled wireless commands to move correctly. It has always been a dream of human being to create machines that behave like humans. Using Remote Control responding accordingly is an important part of this dream. Remote Controlled Structure is an important asset for a robot[11]. Robots are categorized based on the type of control mechanism into two type

2.1.1 Autonomous Robot

These robot have the capacity to think itself and take decision on behalf of human on various aspects. This is due to deep development in the field of AI-artificial intelligence. These robot though are better have not become popular in market due to several reasons:

- Autonomous Robot[12] have decision capabilities but many places like nuclear power plants, decision must be taken by expert persons handling the power plant and not by robot ,else some disasters may occur.
- When a robot is used a spy, it must be handled by military authority as some decision require some harsh decision initially to get benefit later.
- Cost of the making and programming of autonomous robot is very high. In these cases the autonomous robot fail, here we require a manually controlled robots i.e Non-Autonomous Robots.

2.1.2 Non-Autonomous Robot

These Robots[8] have the programming logic to do the desired task but the decision power lies in the hands of the controller handling the robot. Here the interface between the controllers can be made using two method

1. Wired -Here the connection between the controller and robot is maintained using wired interface. These interfaces can be serial or parallel and in the both these techniques, the underlying technology is transmission of the electrical signals, which are sent in form of specific patterns, and the robot to carry out the specific task analyzes these patterns. These signals sent are analyzed by a microcontroller mounted on the robot.

2. Wireless-Here the connection between the controller and robot is maintained using wireless interfaces such as:

   - a. Bluetooth
   - b. Wi-Fi
   - c. Wi-Max
   - d. Zigbee

   The underlying technology is transmission of the signals wirelessly in air by the transmitter, which are captured by the receiver and sent microcontroller mounted on the robot to carry out the decisions. The major benefit of using wireless techniques is that we can receive the live information of the situation to the controller[13].

Chapter 3

Objective And Scope

3.1 Objective

In this project, it is aimed to control a robot using wireless technology. The robot is able to catch and release the snake and to move correctly. We give a direction to robot manually. When the robot gets the direction command, it moves according to command also robot gets catch command it catches the snake correctly. This is a robot whose motions can be controlled by the user by giving specific commands. This is capable of identifying the 6 commands Run, Stop, Left, Right and Back issued by a particular user. What we are aiming at is to control the robot using following Commands.

Robot which can do these basic tasks:

- Move Forward
- Move Backward
- Move Left
- Move Right
- Stop

Robot which can do these main tasks:

- Catch
- Release

The objective of this project is to a develop Robot which is capable to catch snake by taking wireless commands.

3.2 Scope

- Robot and Laptop connectivity using Xbee module and Xbee Chips.
- Embedded C program to control the robot.
- Installation of robotic arm on the robot.
- Installation of wireless camera on the robot.
- Development of snake catcher visual basic application for the laptop to control the robot.

3.3 Out of scope

1. Weight of Snake: Our robot can catch the limited weighted snake up to 2 Kg.
2. Size of Snake: Robot can catch limited size snake up to 20 Cm.
3. Wireless Connection : Limit of wireless connection between Robot and laptop is up to 100 Meters.
4. Wireless Camera: Limit of wireless Camera which is placed on the robot is up to 100 Meters.
5. It is not possible to catch snake if his velocity is more than our limit.

Chapter 4

Requirement Analysis

4.1 Minimum Software Requirements

- Operating System Requirement:

  - Windows XP Or Windows 7.

- Tools and Technology[14]:

  - AVR Studio - Embedded C
  - WinAVR
  - HyperTerminal
  - Visual Basic 4.0
  - TV Home Media3

1. **AVR Studio:** There are number of IDEs (Integrated Development Environment) available for the AVR microcontrollers. There are free IDEs which are based on AVR GCC like AVR Studio from ATMEL and WIN AVR and proprietary IDEs like ICC AVR, Code vision AVR, IAR and KEIL etc. IDEs like ICC AVR and code vision AVR are very simple to use because of their GUI based code generator which gives you generated code. Almost all the proprietary IDEs works as full version for first 45 days and then there code size is restricted to some size. We have used AVR Studio from ATMEL which is feature rich free to IDE for the robot. In this manual we are going to focus on the AVR studio from the ATMEL. It uses WIN

AVR open source C compiler at the back end. It has many attractive features like built-in In-Circuit Emulator and AVR instruction set simulator. After writing and compiling the program it gives [13].

2. **WinAVR:** WinAVR is not just one tool, like many other software names. WinAVR is instead a set of tools, thesetools include avr-gcc (the command line compiler), avr-libc (the compiler library that is essential for avrgcc), avr-as (the assembler), avrdude (the programming interface), avarice (JTAG ICE interface), avr-gdb (the de-bugger), programmers notepad (editor) and a few others. These tools are all compiled for Microsoft Windows and put together with a nice installer program. When referring to the version, you are most of the time referring to the version of the compiler, avr-gcc. For example currently WinAVR includes version 3.3 of avr-gcc[13].

3. **HyperTerminal:** HyperTerminal is program that you can use to connect to other computer ,internet telnet sites, bulletin board system (BBSs), online service and host computer using either your modem or null modem cable HyperTerminal records the messages passed to and from the computer or service on other end of connection . Therefore it can serve as a valuable troubleshooting tool when setting up and using your modem to make sure that your modem is connected properly, you can send command through HyperTerminal and check the result. HyperTerminal has back scroll functionality which allows you to look at receive text that has scroll the screen[12].

4. **Visual Basic:** Visual Basic 2008 is the latest version of Visual Basic launched by Microsoft in the year 2008. It is almost similar to Visual Basic 2005 and but it has added many new features. Visual Basic has gone through many phases of development since the days of BASIC that was built for DOS . BASIC stands for Beginners' All-purpose Symbolic Instruction Code. The program code in Visual Basic resembles the English language. Different software companies had produced many different versions of BASIC for DOS, such as Microsoft QBASIC, QUICK-BASIC, GWBASIC, and IBM BASICA and more. Then, Microsoft launched the first graphical BASIC which was known as Visual Basic Version1 in 1991. It is GUI based and especially developed for MS window. Since then the DOS versions of BASIC were slowly phased out and almost completely replaced by Visual Basic. Visual Basic was initially a functional or procedural programming language until the popular Visual Basic 6. Then, Microsoft decided to make Visual Basic into more powerful object oriented programming language, Visual Basic 2005 was launched with that purpose in mind. Visual Basic 2005 is an object oriented programming language and it was to be taken over by Visual Basic 2008. Visual

Basic 2008 is a full-fledged Object-Oriented Programming (OOP) Language, so it has caught up with other OOP languages such as C++, Java, CSharp and others. However, you don't have to know OOP to learn VB2008. In fact, if you are familiar with Visual Basic 6, you can learn VB2008 effortlessly because the syntax and interface are almost similar. Visual Basic 2008 Express Edition is available free for download from the Microsoft site[11].

4.2 Minimum Hardware Requirements

- Well Configured Laptop
  1. Processor :- Minimum: 1.60GHz Pentium Processor.
  2. RAM :- Minimum: 512MB.
  3. Hard Disk :- Minimum: 40GB.
- Firebird V robot
- Robotic arm
- Xbee chip and module
- TV tuner
- Wireless camera with camera pod

1. **Firebird V Robot[14]:** Fire Bird V ATMEGA2560 technical specification
   - **Microcontroller:**
     (a) Atmel ATMEGA2560 as Master microcontroller (AVR architecture based Microcontroller)
     (b) Atmel ATMEGA8 as Slave microcontroller (AVR architecture based Microcontroller)
   - **Sensors:**
     (a) Three white line sensors (extendable to 7)
     (b) Five Sharp GP2Y0A02YK IR range sensor (One in default configuration)
     (c) Eight analog IR proximity sensors
     (d) Two position encoders (extendable to four)
     (e) Battery voltage sensing
     (f) Current Sensing (Optional)
     (g) Five MaxBotix Ultrasonic Range Sensors (Optional)

- **Indicators:**
  (a) 2 x 16 Characters LCD
  (b) Buzzer and Indicator LEDs
- **Control:**
  (a) Autonomous Control
  (b) PC as Master and Robot as Slave in wired or wireless mode
- **Communication:**
  (a) USB Communication
  (b) Wired RS232 (serial) communication
  (c) Wireless ZigBee Communication (2.4GHZ) (if XBee wireless module is installed)
  (d) Wi-Fi communication (if Wi-Fi module is installed)
  (e) Bluetooth communication (if Bluetooth wireless module is installed)
  (f) Simplex infrared communication (From infrared remote to robot)
- **Dimensions:**
  (a) Diameter: 16cm
  (b) Height: 8.5cm
  (c) Weight: 1100gms
- **Power:**
  (a) 9.6V Nickel Metal Hydride (NiMH) battery pack and external Auxiliary power from battery charger.
  (b) On Board Battery monitoring and intelligent battery charger.
- **Battery Life:**
  (a) 2 Hours, while motors are operational at 75
- **Locomotion:**
  (a) Two DC geared motors in differential drive configuration and caster wheel at front as support
  (b) Top Speed: 24 cm / second
  (c) Wheel Diameter: 51mm
  (d) Position encoder: 30 pulses per revolution

2. **Robotic Arm :** Gripper assembly is used for picking up small objects or holding other robots back side to form chain of robots. Gripping action is actuated using servo motor. Power and the control signal for this servo motor are taken from the S1 servo motor port[14].

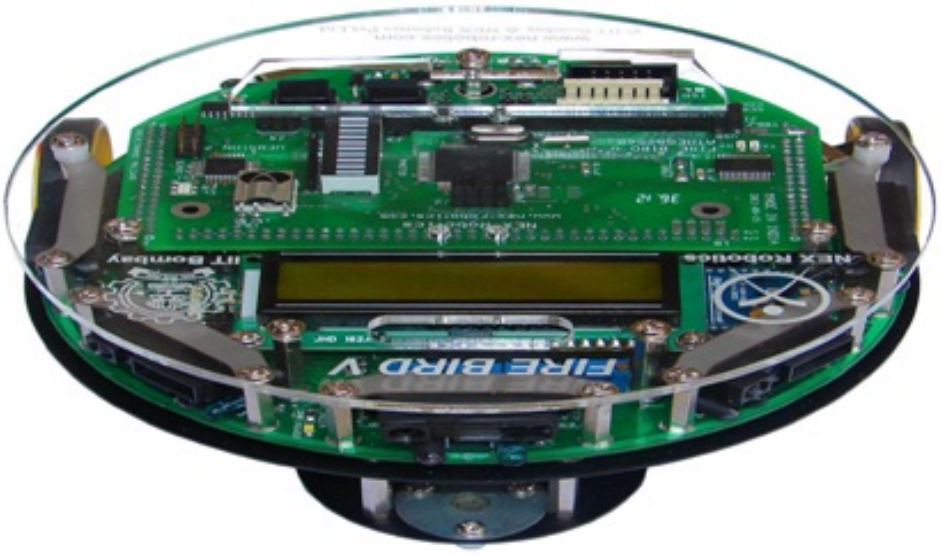

FIGURE 4.1: Firebird V Robot

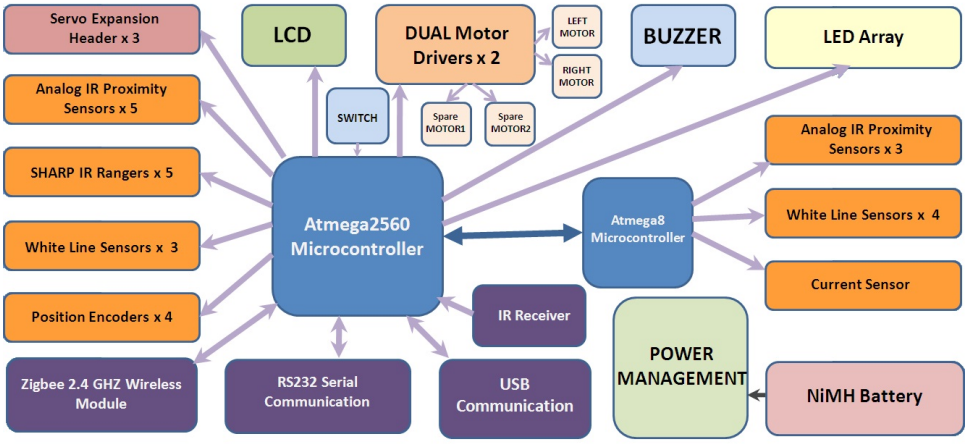

FIGURE 4.2: Black Diagram

- Operating Voltage : 5V
- Grip and Un-grip : 0 to 180 degrees

3. **Xbee chip and Module :** XBee USB wireless adaptor board is used for interfacing any of the series 1 XBee wireless Modules with the PC. Using this USB adaptor board for XBee wireless modules you can Communicate between PC to PC, PC to robot embedded board with ease. You can also use this module to change configuration of the XBee devices using PC via USB port using XCTU Software from Digi. On the PC side this device is treated as the Communication Device Class (CDC) of USB family and it allows the user to treat USB port as a normal serial port (Virtual COM port). XBee USB wireless adaptor board has

five indicator LEDs which shows power, status of the data being transmitted or received, signal strength of the data reception and the association[14].

FIGURE 4.3: Xbee Chip

4. **Wireless Camera**[14] **:** Wireless camera contains following accessories:

   - Wireless camera: Qty. 1;
   - Wireless receiver module: Qty. 1;
   - Antenna for wireless receiver module: Qty. 1;
   - AC Adaptors / SMPS for wireless camera and wireless receiver module: Qty. 2;
   - 9V Battery connector socket for wireless camera: Qty. 1;
   - Audio and Video AV wire: Qty. 1;

FIGURE 4.4: Xbee Chip

Chapter 5

UML Diagram

- **Goals of UML :** The primary goals in the design of the UML were:

  1. Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models. Provide extensibility and specialization mechanisms to extend the core concepts.
  2. Be independent of particular programming languages and development processes.
  3. Provide a formal basis for understanding the modeling language
  4. Encourage the growth of the OO tools market.
  5. Support higher-level development concepts such as collaborations, frameworks, patterns and components.
  6. Integrate best practices

- **Types of UML Diagrams :** There are following types of UML diagrams:

  1. Use case diagram
  2. Sequence diagram
  3. Class diagram
  4. Deployment diagram

- **Modules :**

  1. User
  2. Media Device
  3. Webcam

5.1 Use case Diagram

In the use case diagram actor perform all operation like robot move forward, move backward, move left, move right ,stop and catch as well as release the snake.

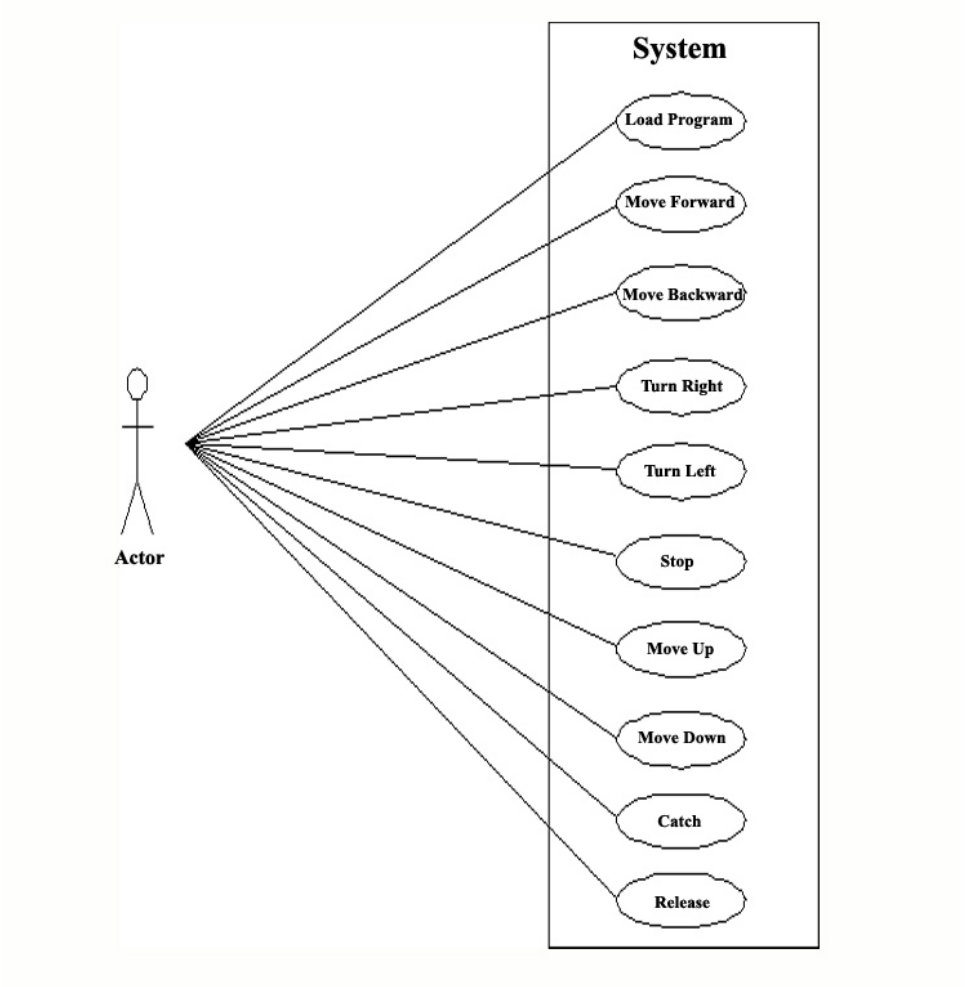

FIGURE 5.1: Use Case Diagram

5.2 Usecase Scenario

Use case	Description
Move Forward	1. Connect One Xbee to the laptop using Xbee module and 2nd Xbee place on the robot. 2. Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. 3. Robot Xbee will send number 8 to the robot and robot will convert that signal number 8 to ASCII value then robot will move forward.
Move Backward	1. Connect One Xbee to the laptop using Xbee module and 2nd Xbee place on the robot. 2. Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. 3. Robot Xbee will send number 2 to the robot and robot will convert that signal number 2 to ASCII value then robot will move backward.
Move Left	1. Connect One Xbee to the laptop using Xbee module and 2nd Xbee place on the robot. 2. Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee.

	3. Robot Xbee will send number 4 to the robot and robot will convert that signal number 4 to ASCII value then robot will move left.
Move Right	1. Connect One Xbee to the laptop using Xbee module and 2nd Xbee place on the robot. 2. Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. 3. Robot Xbee will send number 6 to the robot and robot will convert that signal number 6 to ASCII value then robot will move right.
Stop	1. Connect One Xbee to the laptop using Xbee module and 2nd Xbee place on the robot. 2. Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. 3. Robot Xbee will send number 5 to the robot and robot will convert that signal number 5 to ASCII value then robot will stop.
Catch the Snake	1. Connect One Xbee to the laptop using Xbee module and 2nd Xbee place on the robot. 2. Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. 3. Robot Xbee will send number 1 to the robot and robot will convert that signal number 1 to ASCII value and send to the robotic arm then arm will catch the snake.
Move Arm Up	1. Connect One Xbee to the laptop using Xbee module and 2nd Xbee place on the robot. 2. Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. 3. Robot Xbee will send number 9 to the robot and robot will convert that signal number 9 to ASCII value and send to the robotic arm then arm will move arm at up
Move Arm Down	1. Connect One Xbee to the laptop using Xbee module and 2nd Xbee place on the robot. 2. Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. 3. Robot Xbee will send number 3 to the robot and robot will convert that signal number 3 to ASCII value and send to the robotic arm then arm will move arm at down.

Release the Snake	1. Connect One Xbee to the laptop using Xbee module and 2nd Xbee place on the robot. 2. Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. 3. Robot Xbee will send number 7 to the robot and robot will convert that signal number 7 to ASCII value and send to the robotic arm then arm will release the snake.

5.3 Sequence Diagram

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. If the lifeline is that of an object, it demonstrates a role. Note that leaving the instance name blank can represent anonymous and unnamed instances. Messages, written with horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesnt have to wait for a response. Asynchronous calls are present in multithreaded applications and in message-oriented middleware. Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message (Execution Specifications in UML).

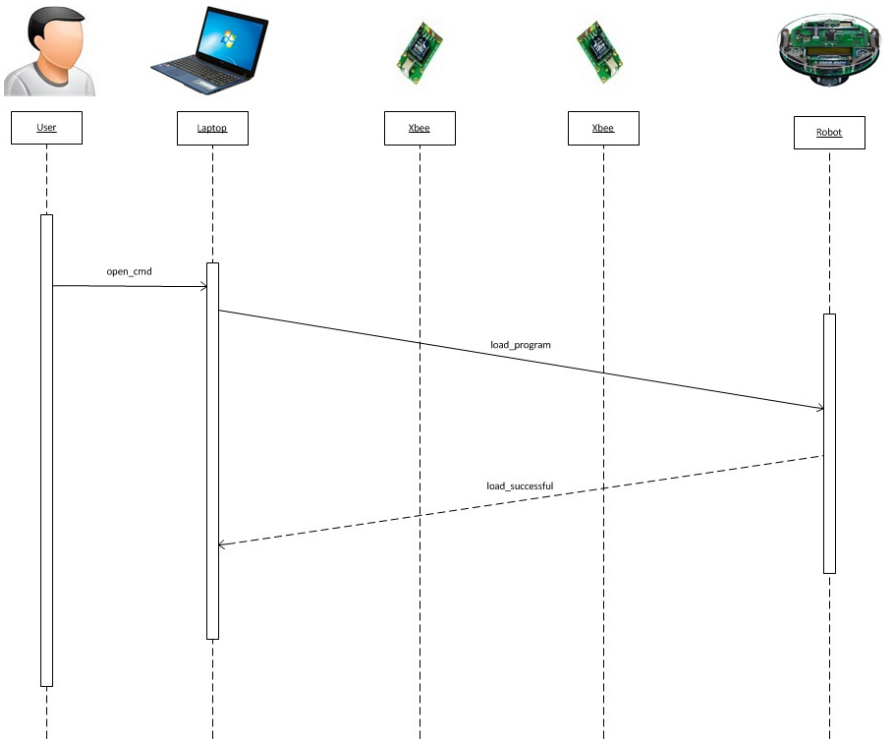

FIGURE 5.2: Sequence Diagram For Load Program On Robot

User open the command prompt then write the Embedded C program on the AVR studio. On AVR studio it require particular frequency(depending upon microcontroller) , set frequency to 14745600 Run the program. it will create one .hex file on the program folder. Then create one new AVRdude folder on C drive and paste that .hex file on that folder. Also paste configuration file(copy from E-yantra CD) on AVRdude folder . After that open command prompt and give the path of AVRdude folder and write one Command :avrdude -c stk500v2 -p m2560 -P NEX-USB-ISP -U flash:w:"program name.hex":i ,

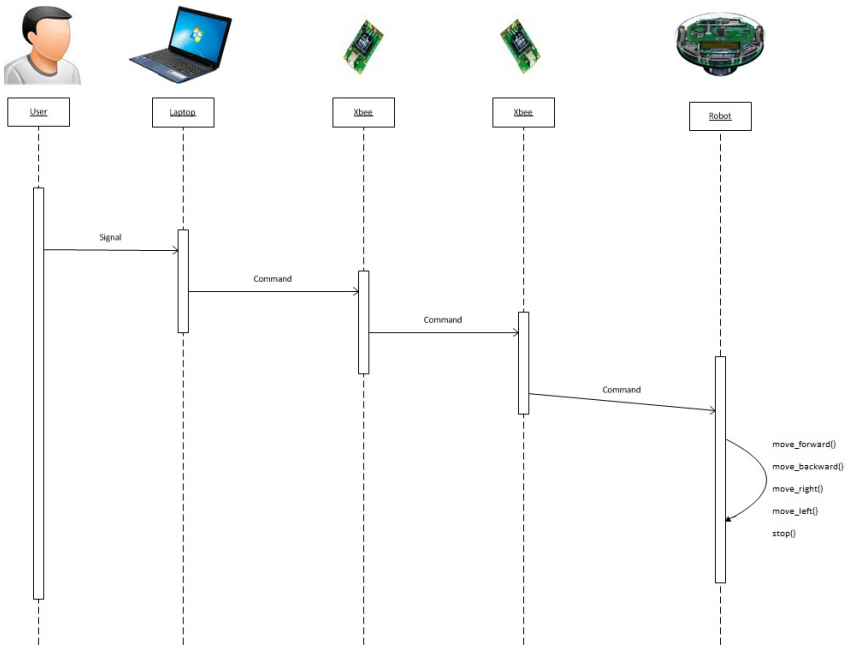

FIGURE 5.3: Sequence Diagram For movement of the Robot at forward, backward left, right direction and stop Robot.

In this fig user send signal to the laptop and laptop send the command to the Xbee2 which placed on the robot after the Xbee2 send the command to the robot after sending the command robot will perform the operation such as Robot at forward, backward left, right direction and stop Robot

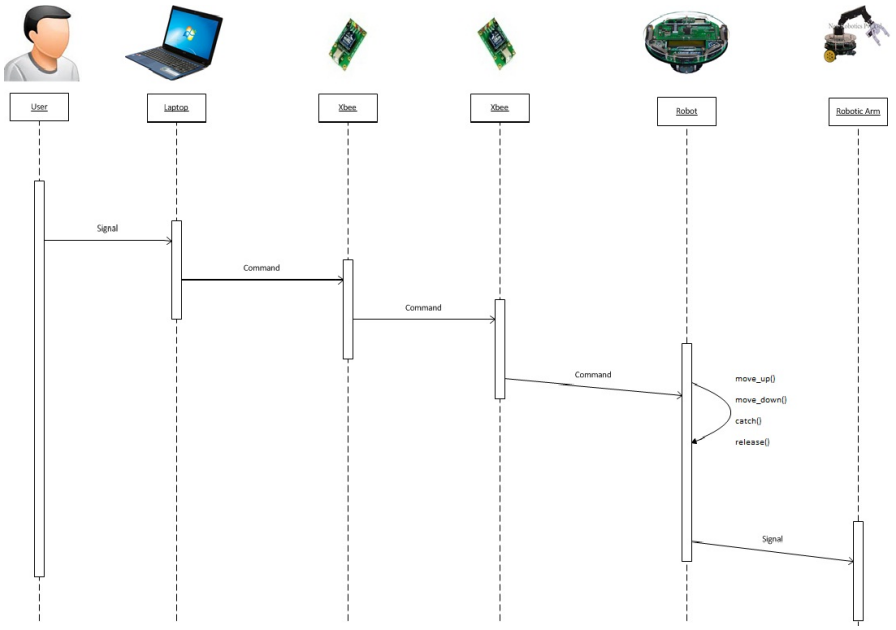

FIGURE 5.4: Sequence Diagram For movement of the Robotic arm at upward, downward direction and catch ,release snake

In this fig user send signal to the laptop and laptop send the command to the Xbee2 which placed on the robot after the Xbee2 send the command to the robot after sending the command robot will perform the operation Robotic arm at upward, downward direction and catch ,release snake

5.4 Class Diagram

A UML class diagram describes the object and information structures used by your application both internally and in communication with its users. It describes the information without reference to any particular implementation. Its classes and relationships can be implemented in many ways, such as database tables, XML nodes, or compositions of software objects .The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

- Class diagram :classes are represented with boxes which contain three parts:

1. The top part contains the name of the class. It is printed in Bold, centered and the first letter capitalized.
2. The middle part contains the attributes of the class. They are left aligned and the first letter is lower case.
3. The bottom part gives the methods or operations the class can take or undertake. They are also left aligned and the first letter is lower case.

- Class: A definition of objects that share given structural or behavioral characteristics.
- Attribute: A typed value attached to each instance of a classifier.
- Operation: A method or function that can be performed by instances of a classifier.

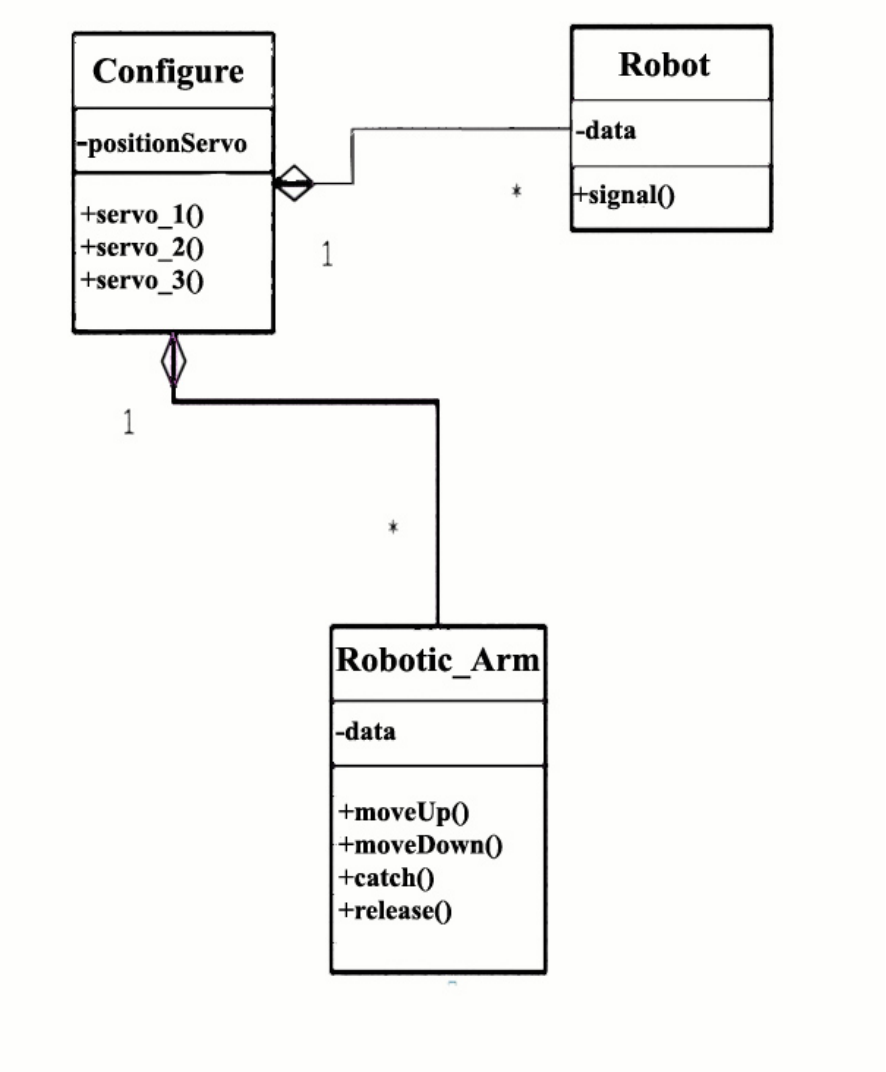

FIGURE 5.5: Class Diagram

5.5 Deployment Diagram

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI). The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

- There are two types of Nodes:

1. Device Node
2. Execution Environment Node

Device nodes are physical computing resources with processing memory and services to execute software, such as typical computers or mobile phones. An execution environment node (EEN) is a software computing resource that runs within an outer node and which itself provides a service to host and execute other executable software elements[14][13].

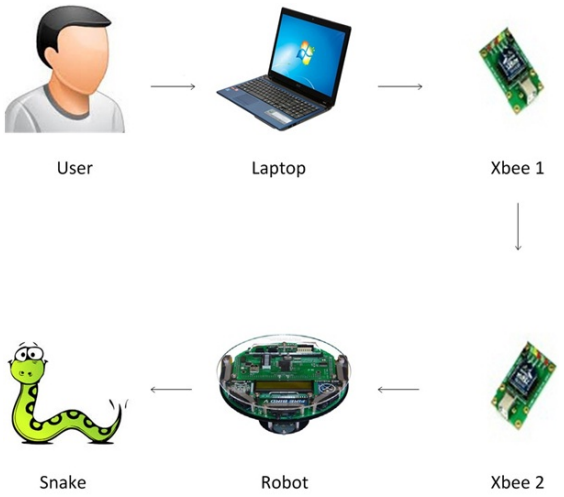

FIGURE 5.6: Deployment Diagram

5.5.1 Camera pod mounting on the Fire Bird V robot[5]

Figure shows the fully assembled servo pod with the wireless camera.

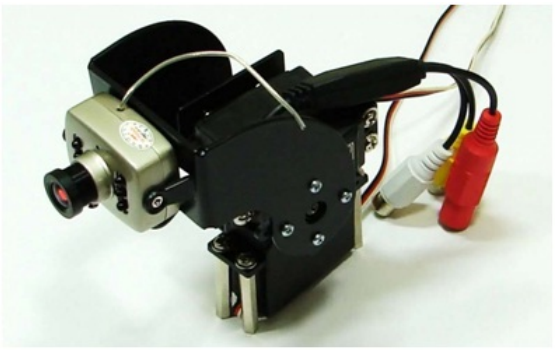

FIGURE 5.7: Camera Pod

5.5.1.1 Camera installation

Step 1: Remove the acrylic top plate from the Robot.

FIGURE 5.8: Camera Installation

Step 2: Mount the camera pod on the Acrylic plate with M3 washer nuts which are fixed on the respective studs.

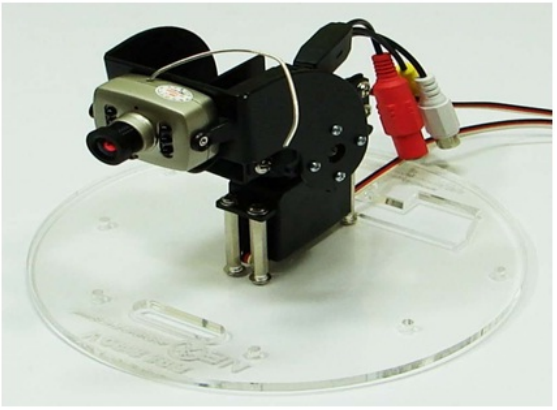

FIGURE 5.9: Camera Pod on Plate

Step 3: Fit the acrylic top plate on the robot as shown in figure. Step 4: Figure shows

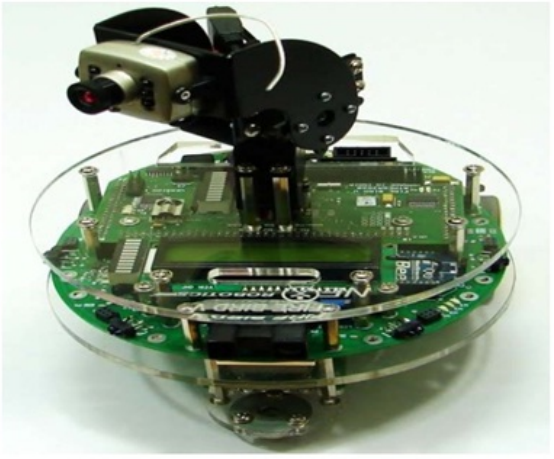

FIGURE 5.10: Camera Installation on Robot

the location of the connectors for the servo motors. Connect Pan Servo motor to S1 and Tilt servo to the S2 connector. Figure shows the actual connections. Notice the white wires direction. If you have servo motor with orange, red and brown wires then while connecting, orient orange wire in the similar position of white wire as shown in figure.

Camera pod	ATMEGA 2560 microcontroller
Tilt servo	OC1A (uC pin 24)
Pan servo	OC1B (uC pin 25)

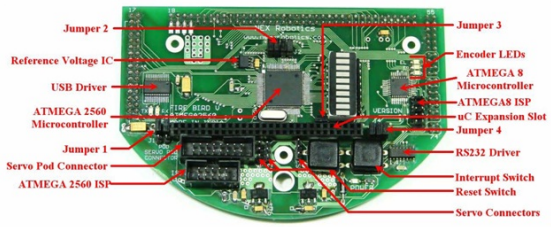

FIGURE 5.11: Main Board Structure

5.5.2 Gripper assembly for Fire Bird V

Gripper assembly is used for picking up small objects or holding other robots back side to form chain of robots.. Gripping action is actuated using servo motor. Power and the control signal for this servo motor are taken from the S1 servo motor port[13].

FIGURE 5.12: Gripper Installation on Robot

5.5.2.1 Servo Connection

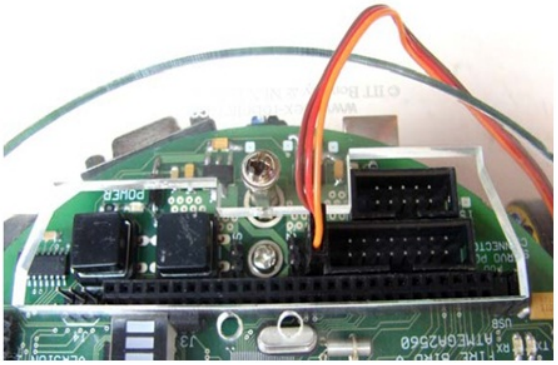

FIGURE 5.13: Servo Motor Pin Connection

5.5.3 Application Example

5.5.3.1 Robot gripper control

In this Application example the grippers servo motor is connected to the S1 servo connector. Servo motor connection is shown in figure 2 and 3. This application example is located in the Accessories Robot gripper control spl case folder in the documentation CD. Upon powering up the robot will un-grip the gripper and move forward for 2.5Sec, pick the object by gripping the object. Move back for 0.5sec, turn left by approx 180 degrees, move forward for 2.5Sec, leave the object by un-gripping the object and move to the start position to pick the next object. This operation is continued, till the robot is powered off[13].

5.5.3.2 Warning

- Power to the servo motor is provided by 5V low drop voltage regulator which is located on the microcontroller board. It can provide only 500mA current. In order to prevent overload on the gripper grip the object with just the necessary force. Do not over grip the object.
- Do not keep servo motor in tight grip position for more than 30 seconds. [13].

Chapter 6

Coding

6.1 Coding standards

General coding standards pertain to how the developer writes code. The SISEPG has come up with a small set of items it feels should be followed regardless of the programming language being used.

6.1.1 Indentation

Proper and consistent indentation is important in producing easy to read and maintainable programs. Indentation should be used to:

- Emphasize the body of a control statement such as a loop or a select statement
- Emphasize the body of a conditional statement
- Emphasize a new scope block

A minimum of 3 spaces shall be used to indent. Generally, indenting by three or four spaces is considered to be adequate. Once the programmer chooses the number of spaces to indent by, then it is important that this indentation amount be consistently applied throughout the program. Tabs shall not be used for indentation purposes.

6.1.2 Inline comments

Inline comments explaining the functioning of the subroutine or key aspects of the algorithm shall be frequently used. See section 4.0 for guidance on the usage of inline comments.

```
Examples:
/* Indentation used in a loop construct. Four spaces are used for indentation. */
for ( int i = 0 ; i < number_of_employees ; ++i )
{
total_wages += employee [ i ] . wages ;
}
// Indentation used in the body of a method.
package void get_vehicle_info ( )
{
System.out.println ( "VIN: " + vin ) ;
System.out.println ( "Make: " + make ) ;
System.out.println ( "Model: " + model ) ;
System.out.println ( "Year: " + year ) ;
}
/* Indentation used in a conditional statement. */
IF ( IOS .NE. 0 )
WRITE ( * , 10 ) IOS
ENDIF
10 FORMAT ( "Error opening log file: ", I4 )
\subsection{Inline Comments}
```

FIGURE 6.1: Code For Indentation

6.1.3 Structured programming

Structured (or modular) programming techniques shall be used. GO TO statements shall not be used as they lead to spaghetti code, which is hard to read and maintain, except as outlined in the FORTRAN Standards and Guidelines.

6.1.4 Classes, subroutines, functions, and methods

Keep subroutines, functions, and methods reasonably sized. This depends upon the language being used. For guidance on how large to make software modules and methods, see section 4.0. A good rule of thumb for module length is to constrain each module to one function or action (i.e. each module should only do one thing). If a module grows too large, it is usually because the programmer is trying to accomplish too many actions at one time. The names of the classes, subroutines, functions, and methods shall have verbs in them. That is the names shall specify an action,

```
e.g. "get_name", "compute_temperature".
```

FIGURE 6.2: Code for Structured Programming

6.1.5 Source files

The name of the source file or script shall represent its function. All of the routines in a file shall have a common purpose.

6.1.6 Variable names

Variable shall have mnemonic or meaningful names that convey to a casual observer, the intent of its use. Variables shall be initialized prior to its first use.

6.1.7 Use of braces

In some languages, braces are used to delimit the bodies of conditional statements, control constructs, and blocks of scope. Programmers shall use either of the following bracing styles:

```
for (int j = 0 ; j < max_iterations ; ++j)
{
/* Some work is done here. */
}
or the Kernighan and Ritchie style:
for ( int j = 0 ; j < max_iterations ; ++j ) {
/* Some work is done here. */
}
```

FIGURE 6.3: Use of Braces

It is felt that the former brace style is more readable and leads to neater-looking code than the latter style, but either use is acceptable. Whichever style is used, be sure to be consistent throughout the code. When editing code written by another author, adopt the style of bracing used. Braces shall be used even when there is only one statement in the control block. For example:

```
Bad:
if (j == 0)
printf ("j is zero.\n");
Better:
if (j == 0)
{
printf ("j is zero.\n");
}
```

FIGURE 6.4: Control Block

6.1.8 Compiler warnings

Compilers often issue two types of messages: warnings and errors. Compiler warnings normally do not stop the compilation process. However, compiler errors do stop the compilation process, forcing the developer to fix the problem and recompile. Compiler and linker warnings shall be treated as errors and fixed. Even though the program will continue to compile in the presence of warnings, they often indicate problems which

may affect the behavior, reliability and portability of the code. Some compilers have options to suppress or enhance compile-time warning messages. Developers shall study the documentation and/or man pages associated with a compiler and choose the options which fully enable the compilers code-checking features.

6.1.9 Actual Work

6.1.9.1 Software availability

In this project we had to use some softwares like AVR studio 4.0 and Win AVR for Writing Embeded C program. These two softwares are core softwares in this project. Nex robotic lab provide these softwares in DVD that we give above path for these softwares. other softwares are freeware so we can download it from internet, i.e. Hyperterminal, Visual Studio,TV tuner setup. Path of softwares in DVD : Open CD e Yantra DVDFire Bird V ATMEGA2560 Robot Software and Drivers.

6.1.9.2 Problems facing before implementation

All the hardware we are using that is Created by Nex Robotic lab Under the IIT Bombay. So they do not provide any information on internet. For this reason we are facing so many problems, like installation and use of all hardware parts. We dint know how to use any hardware in this project. But under the guidance of Mr. M. I. Mullani and Venkatesh Shankar we had solve these problems.

6.1.9.3 Rasberripi module

At the Starting of that project we Having RasberriPi Module but dont know what is RasberriPi Module and how to use it.rather than we want to use rasberripi module. After research we have to know that, RasberriPi is actually use for controlling any device using Linux OS and we can also use this device as a small computer(Linux Based). But that time we want to use Windows OS because some softwares are not supporting on linux Os that purpose we cancelled the use of this Module.

6.1.9.4 Xbee module and its connection

Another issue is the Xbee Module. Same problem come for this module that is facing problem in use of rasberri V. At the Starting of that project we Having Xbee Module but dont know what is Xbee Module and how to use it. rather than we want to use Xbee

module. After research we have to know that, Xbee is actually use for communication between two hardwares. using Xbee module we can send signal from one Xbee to another Xbee. That time we was unsuccessful in connection because on internet many Xbee modules are available but installation for every xbee is different. Long time we successful in connection.

6.1.9.5 Softwares used for this project[13]

Stepwise Installation :

- **AVR Studio 4.0**
- **WinAVR**

6.1.9.6 IDE installation

Since AVR studio uses WIN AVR compiler at the back end we need to install WIN AVR first before installing AVR studio. By doing so AVR Studio can easily detect the AVRGCC plug-ins.

6.1.9.7 Installing WIN AVR

Copy WIN AVR 2009-03-13 from the Software and Drivers folder of the documentation CD on the PC and click on WinAVRxxxx.exe file.

The following window with WIN AVR installation package will open.

- Choose language asEnglish.
- Click next in the WIN AVR setup wizard.
- Press I Agree after going through license agreement.
- Make sure that you have to select the drive on which operating system is installed.
- Select all the components and press Install.
- Click Finish to complete WIN AVR installation.

6.1.9.8 Installing AVR studio[13]

- Go to Software and Drivers folder from the documentation CD, copy folder AVR Studio 4.17 on the PC and click on AvrStudio417Setup.exe to start the installation process.

- Click on Run
- Click Next to start installation of AVR Studio 4
- After clicking Next, go through the license agreement. If it is acceptable then click Next
- Now choose the destination drive. Select the same drive in which your operating system and WINAVR is installed.
- Select for the Install / upgrade Jungo USB Driver to support In System Programming (ISP) by AVRISP mkII
- Important: If Install / upgrade Jungo USB Driver is not selected then AVRISP mkII programmer will not work with the AVR Studio.
- Click Next to start the installation process.
- Click Finish to complete the installation process.

6.1.9.9 Setting up project in AVR studio

AVR studio is an Integrated Development Environment (IDE) for writing and debugging AVR applications. As a code writing environment, it supports includes AVR Assembler and any external AVR GCC compiler in a complete IDE environment.
AVR Studio gives two main advantages:

1. Edit and debug in the same application window. Faster error tracking.
2. Breakpoints are saved and restored between sessions, even if codes are edited.

6.1.9.10 Use of AVR studio[13]

- Open AVR Studio. If any project is running it can be closed by clicking on Project in the
- menu bar and select Close Project.
- To create a new project, click on Project in the menu bar and select New Project.
- Select Project Type as AVR GCC. Type project name in the Project name window.
- In this case it is buzzer test. Also check on Create initial file and Create folder check box. This will create all the files inside the new folder. In the Location window select the place where would like to store your project folder and then click Next.

- Select debug platform and Device. In this case we have selected AVR simulator and ATMEGA2560 microcontroller and click finish.
- Now we are almost ready to write our first code. Before we start coding we will check other setting to make sure that they are set properly.
- Open Project menu and click on the Configuration option.
- In the Project Option General tab will open.
- Select device as ATMEGA2560 and frequency (Crystal Frequency) as 14.7456MHz i.e. 14745600Hz. Set the optimization level be at-O0.

The levels of optimization are:

- -O0 No optimization. This is the same as not specifying any optimization.
- -01 Optimize. Reduces code size and execution time without performing any optimizations that take a great deal of compilation time.
- -O2 Optimize even more. avr-gcc performs almost all optimizations that don't involve a space-time tradeoff.
- -O3 Optimize yet more. This level performs all optimizations at -O2 along with finlinefunctions and -frename-registers.
- -Os Optimize for size. Enables all -O2 optimizations that don't increase code size. It also
- performs further optimizations designed to reduce code size.
- For more information on optimization, see the 'man' pages for avr-gcc.
- Important: During the coding choose appropriate optimization option. If you feel that code is not working properly as it should be then turn off all optimization by selecting optimization
- option as -O0. Once you know that your code is properly working then you can incrementally increase optimization level.

6.1.9.11 How to load program on the robot

1. Write the Embedded C program on the AVR studio.
2. On AVR studio it require particular frequency(depending upon microcontroller) , set frequency to 14745600
3. Run the program.
4. it will create one .hex file on the program folder.
5. Then create one new AVRdude folder on C drive and paste that .hex file on that folder. Also paste configuration file(copy from E-yantra CD) on AVRdude folder .

6. After that open command prompt and give the path of AVRdude folder and write one Command :avrdude -c stk500v2 -p m2560 -P NEX-USB-ISP -U flash:w:"program name.hex":i

- **HyperTermial**

  For installation just click on .exe file then it will automatically installed after installatiion setting as follows[12]:

  - On the file menu, click New Connection
  - In the Name box, type a name that describe the connection.
  - In the Icon box, click the appropriate icon and then click OK.
  - In the Connect To dialog box , enter the information for the phone number you want to dial and click OK.
  - If you are using modem and want to change your dialing properties , click Dialing Properties, make your changes and then Click OK.
  - To dial the call, Click Dial
  - If you use laptop port connection then select port number and click OK

6.2 Review of code

```
#include<avr/io.h>
#include<avr/interrupt.h>
#include<util/delay.h>}

unsigned char data; //to store received data from UDR1
void servo1_pin_config (void)
{
DDRB  = DDRB | 0x20;  //making PORTB 5 pin output
 PORTB = PORTB | 0x20; //setting PORTB 5 pin to logic 1
}
//Configure PORTB 6 pin for servo motor 2 operation
void servo2_pin_config (void)
{
DDRB  = DDRB | 0x40;  //making PORTB 6 pin output
 PORTB = PORTB | 0x40; //setting PORTB 6 pin to logic 1
}
//Configure PORTB 7 pin for servo motor 3 operation
void servo3_pin_config (void)
{
DDRB  = DDRB | 0x80;  //making PORTB 7 pin output
 PORTB = PORTB | 0x80; //setting PORTB 7 pin to logic 1
}


//Function to rotate Servo 1 by a specified angle in the multiples of 1.86 degrees
void servo_1(unsigned char degrees)
{
floatPositionPanServo = 0;
PositionPanServo = ((float)degrees / 1.86) - 35.0;
 OCR1AH = 0x00;
 OCR1AL = (unsigned char) PositionPanServo;
}
void servo_11(unsigned char degrees)
{
floatPositionPanServo = 0;
PositionPanServo = ((float)degrees / 1.86) + 35.0;
 OCR1AH = 0x00;
 OCR1AL = (unsigned char) PositionPanServo;
}
//Function to rotate Servo 1 by a specified angle in the multiples of 1.86 degrees
void servo_111(unsigned char degrees)
{
floatPositionPanServo = 0;
PositionPanServo = ((float)degrees / 1.86) - 35.0;
 OCR1AH = 0x00;
 OCR1AL = (unsigned char) PositionPanServo;
}
void servo_112(unsigned char degrees)
{
floatPositionPanServo = 0;
PositionPanServo = ((float)degrees / 1.86) + 35.0;
 OCR1AH = 0x00;
```

```
 OCR1AL = (unsigned char) PositionPanServo;
}
//Function to rotate Servo 2 by a specified angle in the multiples of 1.86 degrees
void servo_2(unsigned char degrees)
{
floatPositionTiltServo = 0;
PositionTiltServo = ((float)degrees / 1.86) - 35.0;
 OCR1BH = 0x00;
 OCR1BL = (unsigned char) PositionTiltServo;
}
void servo_22(unsigned char degrees)
{
floatPositionTiltServo = 0;
PositionTiltServo = ((float)degrees / 1.86) + 35.0;
 OCR1BH = 0x00;
 OCR1BL = (unsigned char) PositionTiltServo;
}
void servo_222(unsigned char degrees)
{
floatPositionTiltServo = 0;
PositionTiltServo = ((float)degrees / 1.86) - 35.0;
 OCR1BH = 0x00;
 OCR1BL = (unsigned char) PositionTiltServo;
}
void servo_223(unsigned char degrees)
{
floatPositionTiltServo = 0;
PositionTiltServo = ((float)degrees / 1.86) + 35.0;
 OCR1BH = 0x00;
 OCR1BL = (unsigned char) PositionTiltServo;
}
//Function to rotate Servo 3 by a specified angle in the multiples of 1.86 degrees
void servo_3(unsigned char degrees)
{
floatPositionServo = 0;
PositionServo = ((float)degrees / 1.86) + 35.0;
 OCR1CH = 0x00;
 OCR1CL = (unsigned char) PositionServo;
}
void servo_33(unsigned char degrees)
{
floatPositionServo = 0;
PositionServo = ((float)degrees / 1.86) - 35.0;
 OCR1CH = 0x00;
 OCR1CL = (unsigned char) PositionServo;
}



void servo_1_free (void) //makes servo 1 free rotating
{
 OCR1AH = 0x03;
 OCR1AL = 0xFF; //Servo 1 off
}
void servo_2_free (void) //makes servo 2 free rotating
```

```
{
 OCR1BH = 0x03;
 OCR1BL = 0xFF; //Servo 2 off
}
void servo_3_free (void) //makes servo 3 free rotating
{
 OCR1CH = 0x03;
 OCR1CL = 0xFF; //Servo 3 off
}


//Function To Initialize all The Devices
voidinit_devices()
{
cli(); //Clears the global interrupts
port_init();  //Initializes all the ports
 uart0_init(); //Initailize UART1 for serial communiaction

  timer1_init();
sei();   //Enables the global interrupts
}
voidbuzzer_pin_config (void)
{
 DDRC = DDRC | 0x08;            //Setting PORTC 3 as outpt
 PORTC = PORTC & 0xF7;          //Setting PORTC 3 logic low to turnoff buzzer
}
//Function to initialize ports
voidport_init()
{
        motion_pin_config();
        buzzer_pin_config();
        servo1_pin_config(); //Configure PORTB 5 pin for servo motor 1 operation
 servo2_pin_config(); //Configure PORTB 6 pin for servo motor 2 operation
 servo3_pin_config(); //Configure PORTB 7 pin for servo motor 3 operation
}


SIGNAL(SIG_USART0_RECV)         // ISR for receive complete interrupt
{
unsigned char i = 0;
        data = UDR0;            //making copy of data from UDR0 in 'data' variable
        UDR0 = data;            //echo data back to PC
                if(data == 0x38) //ASCII value of 8
                {
                        PORTA=0x06;  //forward
                }
                if(data == 0x32) //ASCII value of 2
                {
                        PORTA=0x09; //back
                }

                if(data == 0x34) //ASCII value of 4
                {
                        PORTA=0x05;  //left
                }
```

```
if(data == 0x36) //ASCII value of 6
{
        PORTA=0x0A; //right
}
if(data == 0x35) //ASCII value of 5
{
        PORTA=0x00; //stop
        servo_1_free();
        servo_2_free();
        servo_3_free();
}
if(data == 0x31) //ASCII value of 1
{
        for (i = 0; i <2; i++)
{
        servo_1(i);
        _delay_ms(30);
        servo_1_free();
}
}
if(data == 0x33) //ASCII value of 3
{
for (i = 0; i <2; i++)
{
servo_2(i);
_delay_ms(30);
 servo_2_free();
}
}
if(data == 0x31) //ASCII value of 1
{
for (i = 0; i <2; i++)
{
servo_1(i);
_delay_ms(30);
servo_1_free();
}
}
if(data == 0x37) //ASCII value of 1
{
        for (i = 0; i <2; i++)
{
servo_11(i);
_delay_ms(30);
servo_1_free();
}
}
if(data == 0x39) //ASCII value of 3
{
for (i = 0; i <2; i++)
{
servo_22(i);
_delay_ms(30);
servo_2_free();
}
```

```
                }
                if(data == 0x37) //ASCII value of 1
                {
                for (i = 0; i <2; i++)
                {
                servo_11(i);
                _delay_ms(30);
                servo_1_free();
                }
        }
}


//Main Function
int main(void)
{
        init_devices();
        while(1);
}
```

Chapter 7

Testing

7.1 Static vs. Dynamic testing

There are many approaches to software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing is often implicit, as proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100Static testing involves verification, whereas dynamic testing involves validation. Together they help improve software quality. Among the techniques for static analysis, mutation testing can be used to ensure the test-cases will detect errors which are introduced by mutating the source code.

7.1.0.12 The box approach

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

7.2 White-box testing

White-box testing (also known as clear box testing, glass box testing, and transparent box testing and structural testing) tests internal structures or workings of a program,

as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a systemlevel test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements. Techniques used in white-box testing include:

- API testing (application programming interface) testing of the application using public and private APIs
- Code coverage creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
- Fault injection methods intentionally introducing faults to gauge the efficacy of testing strategies
- Mutation testing methods
- Static testing methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Code coverage as a software metric can be reported as a percentage for:

- Function coverage, which reports on functions executed
- Statement coverage, which reports on the number of lines executed to complete the test

100 percent statement coverage ensures that all code paths, or branches (in terms of control flow) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly

7.3 Black-box testing

FIGURE 7.1: Black box testing

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing. Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional. Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations. One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight."Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested. This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

7.4 Unit testing

Unit testing is the testing of individual hardware or software units or groups of related units. Using white box testing techniques, testers verify that the code does what it is intended to do at a very low structural level. Unit testing focuses verification effort on the smallest unit of software designthe Software component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered errors is limited by the constrained scope established for unit testing. The unit test is white-box oriented, and the step can be conducted in parallel for multiple components. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

7.5 Integration testing

Integration testing is testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them. The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing Bottom-up Integration testing and Top- Down Integration testing.

- Bottom-up integration testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
- Top-Down integration testing, the highest-level modules are tested first and progressively lower-level modules are tested after that. In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing.

7.6 Validation testing

Software validation testing is essentially a task carried out by a software tester. The aim is to check, if the software has been made in lines with the requirements of the client. While verification is a quality control process, quality assurance process carried out before the software is ready for release is known as validation testing. Its goals are to validate and be confident about the software product or system, that it fulfills the requirements given by the customer. The acceptance of the software from the end

customer is also its part. Often the testing activities are introduced early in the software development life cycle. The two major areas when it should take place are in the early stages of software development and towards the end, when the product is ready for release. In other words, it is acceptance testing which is a part of validation testing.

7.7 System testing

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. This type of testing is performed by a specialized testing team.

7.7.0.13 Why is system testing so important?

- System Testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment which is very close to the production environment where the application will be deployed.
- System Testing enables us to test, verify and validate both the business requirements as well as the Applications Architecture.

7.8 Actual testing of project

7.8.0.14 Unit testing

Test Step	Test Case	Expected Result	Result
Switch on Robot	Robot is Starting	Robot is Starting	Pass
Open Prompt	Terminal is opening	Terminal is opening	Pass
Press Button 8	forward direction	Robot is moving at forward direction	Pass
Press Button 2	backward direction	Robot is moving at backward direction	Pass
Press Button 4	left direction	Robot is moving at left direction	Pass
Press Button 6	right direction	Robot is moving at right direction	Pass
Press Button 5	Stopped	Robot is Stopped	Pass
Press Button 3	Arm is moving down	Arm is moving at down direction	Pass
Press Button 9	Arm is moving up	Arm is moving at up direction	Pass
Press Button 1	Arm is catching	Arm is catching snake	Pass
Press Button 7	Arm is releasing	Arm is releasing snake	Pass

7.8.1 Snap shots

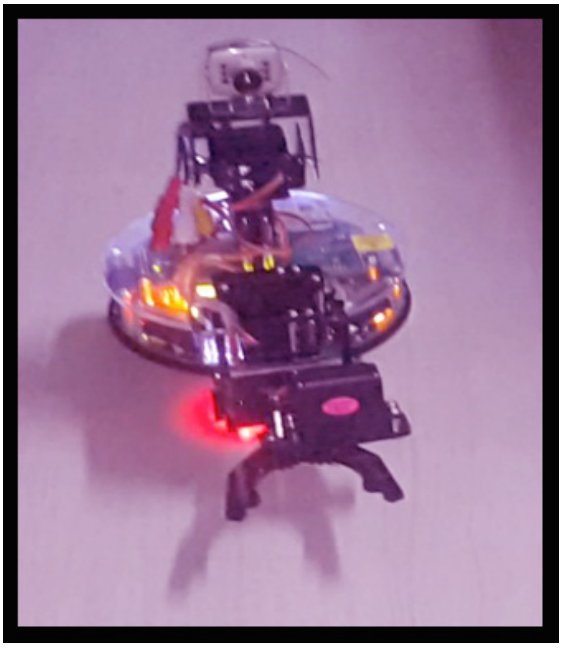

FIGURE 7.2: Starting snapshot

FIGURE 7.3: Snapshot for robot move at right direction

Connect One Xbee to the laptop using Xbee module and 2ndXbee place on the robot.Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. Robot Xbee will send number 6 to the robot and robot will convert that signal number 6 to ASCII value then robot will move right.

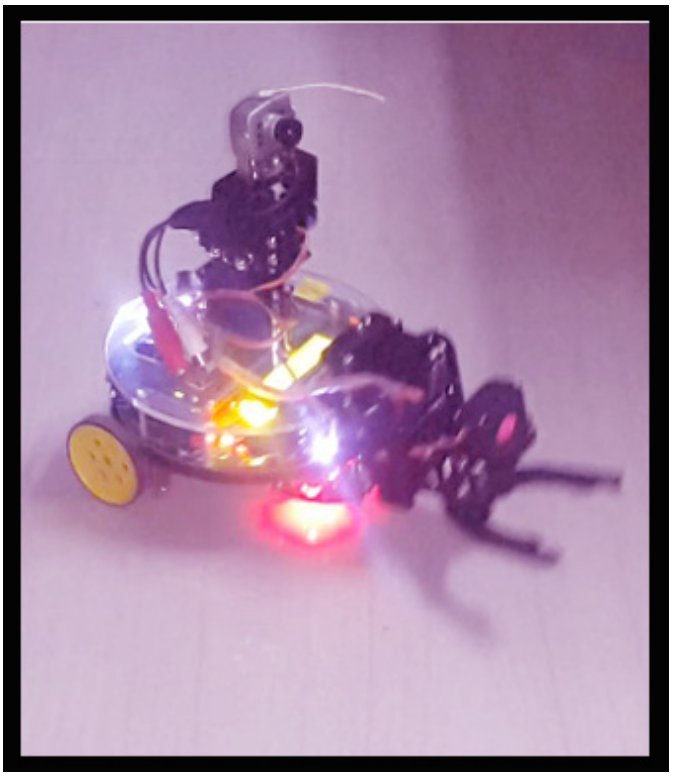

FIGURE 7.4: Snapshot for robot move at left direction

Connect One Xbee to the laptop using Xbee module and 2ndXbee place on the robot.Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee.Robot Xbee will send number 4 to the robot and robot will convert that signal number 4 to ASCII value then robot will moveleft.

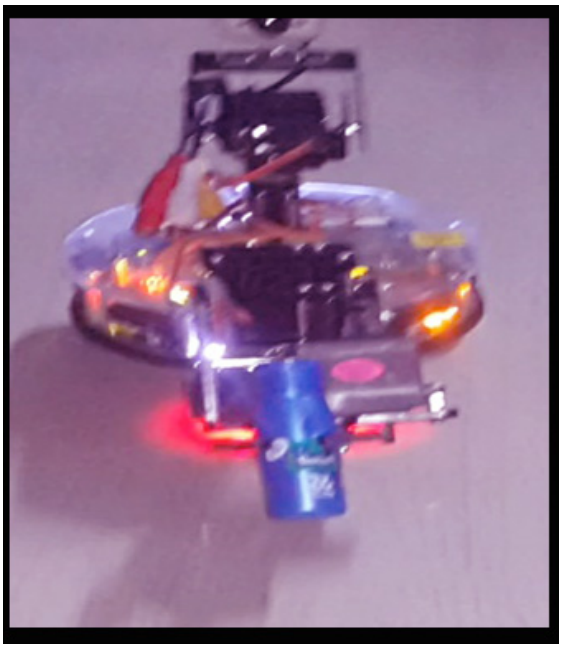

FIGURE 7.5: Snapshot for robot catches object

Connect One Xbee to the laptop using Xbee module and 2ndXbee place on the robot.Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. Robot Xbee will send number 1 to the robot and robot will convert that signal number 1 to ASCII value and send to the robotic arm then arm will catch the snake.

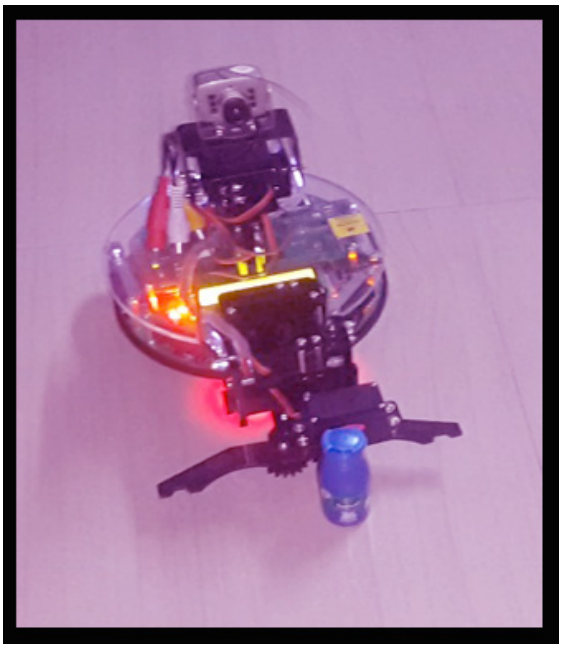

FIGURE 7.6: Snapshot for robot releases object

Connect One Xbee to the laptop using Xbee module and 2ndXbee place on the robot.Open HyperTerminal Application and send signal from laptop Xbee to robot Xbee. Robot Xbee will send number 7 to the robot and robot will convert that signal number 7 to ASCII value and send to the robotic arm then arm will release the snake.

Chapter 8

Conclusion

The robotics- the word has deep meaning. Robotics in turn takes the scale of development by the employing various branches tools, mechanism and performs a wide variety of functions for the benefit of mankind from this whole ocean of robotics for final year projects, we develop Snake Catcher Robot. This robot does all basic functionalities such as robot move forward ,backward, right, left directions; move robotic arm at up and down direction; catch and release the object ; video transmission in its Xbee ranges of operation. This robot does not point out specific application but as whole try to convey the message of cost effective building of snake catcher robot with all capabilities as wired robot can have. Thus user can control a robot using Laptop. The robot is able to understand basic wireless commands to move correctly. It has always been a dream of human being to create machines that behave like humans. Using wireless technology responding accordingly is an important part of this dream. Wireless Structure is an important asset for a robot.

Bibliography

[1] Distributed in the U.S. and Canada. Book of c programming for embedded system by publishers group west 1700 fourth street berkeley, ca 94710 isbn 1-929629-04-4.

[2] IN 46290 USA Howard Hawhee, Senior Author Corby Jordan Richard Hundhausen Felipe Martins Thomas Moore of Email: certification@mcp.com Mail: Mary Foote Executive Editor Certification New Riders Publishing 201 West 103rd Street Indianapolis. Superbot:complete reference of visual basic 6.0.

[3] nationalgeography. www.nationalgeography.comfile/20e2809320nat20geo20tv20blogs files activityi.htm.

[4] http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?reload=trueandpunumber=8860. kinematics, dynamics, control, and simulation of robots and intelligent machines and systems.2013.

[5] http://www.ercim.eu/publication/Ercim News/enw55/nuechter.html2013 IEEE International Conference on Robotics and 2013 http://www.icra2013.org/?pageid=69 Automation Karlsruhe, May 6 10. 2013 ieee international conference on robotics and automationkarlsruhe.

[6] http://www.ercim.eu/publication/Ercim News/enw55/nuechter.html. Kurt 3d: An autonomous mobile robot for modelling the world in 3d.

[7] ISER2000 Shigeo Hirose. Super mechano-system : New perspective for versatilerobotic system.

[8] Aaron Peck Howie Choset JustineRembisz Philip Gianfortoni Allison Naaktgeboren ICRA2007 Kevin Lipkin, Issac Brown. Differential andpiecewise differentiable gaits for snake robots.

[9] http://en.wikipedia.org/wiki/Xbox 360 Wireless Racing Wheel. Microsofts xbox 360 racing wheel.

[10] S Grimmer H Wagner ASeyfarth, H Geyer and M Gunther. Intelligence by mechanics.

[11] Snake prototype S7. http://www.snakerobots.com/s7.html.

[12] Mark Moll BehnamSalemi and Wei-Min Shen 2006 IEEE:RSJ Intl. Conf. on. Superbot: A deployable,multi functional, and modular self-reconfigurable robotic system intelligent robots and system.

[13] www.eyantra.com. http: qa.e-yantra.org dvd drive/fire bird v atmega 2560 robot/-manuals and application notes/manuals.

[14] www.nexrobotics.co.in.. nexrobotics.