"MICROCONTROLLER BASED PC CONTROLLED ROBOT FOR DETECTING HUMAN PRESENCE"

By

Malaya Tripathi (111058)

Anchal Gupta (111130)

Neha Jaiswal (111132)

Under the supervision of

Mrs. Vanita Rana

May-2015

Dissertation submitted in partial fulfilment

of the requirement for the Degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT, SOLAN - 173234, (H.P.) INDIA

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
(Established under the Act 14 of Legislative Assembly of Himachal Pradesh)
Waknaghat, P.O. DomeharBani. Teh. Kandaghat, Distt. Solan- 173234(H.P.)
Phone: 01792-245367, 245368, 245369
Fax- 01792-245362

CERTIFICATE

This is to certify that the work titled **"Microcontroller Based PC Controlled Robot For Detecting Human Presence"** submitted by **"Mr. Malaya Tripathi, Ms. Anchal Gupta and Ms. Neha Jaiswal"** in the partial fulfilment of the degree of Bachelor of Technology (ECE) of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not yet been submitted partially or wholly to any other university or institution for the award of this or any other degree or diploma.

Date:

Mrs. Vanita Rana
(Assistant Professor)
Department of Electronics and Communication Engineering
Jaypee University of Information Technology (JUIT)
Waknaghat, Solan - 173234, (H.P.) India
(Supervisor)

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
(Established under the Act 14 of Legislative Assembly of Himachal Pradesh)
Waknaghat, P.O. DomeharBani. Teh. Kandaghat, Distt. Solan- 173234(H.P.)
Phone: 01792-245367, 245368, 245369
Fax- 01792-245362

DECLARATION

We hereby declare that the work reported in the B. Tech thesis entitled "**Microcontroller Based PC Controlled Robot For Detecting Human Presence**" submitted by "**Mr. Malaya Tripathi, Ms. Anchal Gupta and Ms. Neha Jaiswal**" at Jaypee University of Information Technology, Waknaghat is an authentic record of our work carried out under the supervision of **Mrs. Vanita Rana**. This work has not been submitted partially or wholly to any other university or institution for the award of this or any other degree or diploma.

Mr. Malaya Tripathi (111058)
Ms. Anchal Gupta (111130)
Ms. Neha Jaiswal (111132)
Department of Electronics and Communication Engineering
Jaypee University of Information Technology (JUIT)
Waknaghat, Solan – 173234, (H.P.) India

ACKNOWLEDGEMENT

We feel to convey our indebtedness to all those who helped us to reach our goal. We take this opportunity to express our profound gratitude and deep regards to our guide Mrs. Vanita Rana for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her at different steps shall carry us a long way in the journey of life on which we are about to embark. We are obliged to all faculty members of JUIT, for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our project.

Contents

S. No. Topic **Page No.**

1 INTRODUCTION 11

1.1 Block Diagrams……...12

1.1.1 Control Room Module……...12

1.1.2 Robot Module……12

2 TECHNOLOGY BACKGROUND 13

2.1 Arduino 1.6.3……..13

2.2 X-CTU……14

3 PROPOSED HARDWARE 16

3.1 Arduino UNO……16

3.1.1 Technical Specifications……17

3.1.2 Power Pins……..17

3.1.3 Digital I/O Pins……...18

3.1.4 Analog Input Pins……...18

3.1.5 Physical Characteristics……..19

3.1.6 USB Over Current Protection……19

3.1.7 How to use Arduino?.....19

3.1.8 Functions used in Arduino……..19

3.2 ZigBee……22

3.2.1 XBee……....24

3.2.2 Communication Modes……...25

3.2.3 XBee Configuration……26

3.2.4 XBee Series 2 (ZigBee Mesh)……28

3.2.5 XBee Pin Description……...28

3.2.6 XBee Series 2 ZB Features……29

3.3 XBee USB Adaptor Board……..30

3.4 Arduino XBee Shield……..31
3.5 Temperature Sensor……32
3.5.1 LM35……...32
3.6 PIR Sensor……...33
3.6.1 Basic Principle……34
3.6.2 Differential Detection……..34
3.6.3 How does PIR Sensor work?.....35
3.6.4 Changing Pulse Time and Timeout Length……..37
3.6.5 Technical Specifications……37
3.7 Ultrasonic Sensor……....38
3.7.1 Measurement Principle of Ultrasonic Sensor……..38
3.7.2 HC-SR04……39
3.7.3 Electrical Specifications of HC-SR04……40
3.8 DC Motors……...40
3.8.1 General Specifications……41
3.8.2 Transistor Circuit for Interfacing Motor with Arduino……...42

4 METHODOLOGY 44

4.1 Module 1: Mechanical Construction of Robot……44
4.2 Module 2: XBee Configuration Settings……45
4.2.1 PC Settings……..45
4.2.2 Modem configuration……...46
4.3 Module 3: Sensor Testing……...47
4.3.1 PIR Sensor Testing……...47
4.3.2 Temperature Sensor Testing……....48
4.4 Module 4: PC Controlled Wireless Robot……..48
4.5 Module 5: Assembling all the Sensors and Final Implementation…….50
4.6 Problems Faced and Challenges……57

5 CONCLUSION AND FUTURE SCOPE 58

5.1 Conclusion……...58
5.2 Future Scope……...59

6 REFERENCES 60

List of Figures

S. No. Topic **Page No.**

1.1 Control Room Module……12
1.2 Robot Module……..12

2.1 Arduino Software……13
2.2 X-CTU Software……..14

3.1 Arduino UNO……...16
3.2 ZigBee Technology……..23
3.3 XBee S2……....24
3.4 XBee Pin Configuration……...28
3.5 XBee USB Adaptor Board……...30
3.6 Arduino XBee Shield……...31
3.7 Temperature Sensor……32
3.8 LM35……....32
3.9 PIR Sensor……....33
3.10 Differential Detection……...34
3.11 Working of PIR Sensor……35
3.12 Fresnel Lens……36
3.13 Overview of PIR Sensor……...37
3.14 Ultrasonic Sensor……38
3.15 Principle of Ultrasonic Sensor……38
3.16 HC-SR04……..39
3.17 Timing Diagram Showing Working of HC-SR04……...40
3.18 DC Gear Motor.....41

3.19 Transistor Circuit for Motor Interfacing……..42
3.20 Breadboard Layout of Motor Interfacing Circuit…….43

4.1 Fixing DC Motors and Wheels on the Metal Chassis…….....45
4.2 XBee Router PC Settings…….45
4.3 XBee Coordinator PC Settings…….45
4.4 Router Configuration Settings……..46
4.5 Coordinator Configuration Settings…….....46
4.6 PIR Sensor Testing Circuit……...47
4.7 LM35 Testing Circuit……...48
4.8 Top View of Robot…….....55
4.9 Front View of Robot……56
4.10 Terminal at Personal Computer……...56

List of Tables

S. No. **Topic** **Page No.**

3.1 XBee Configuration Settings……16

4.1 PIR Sensor Testing Circuit Readings……17

4.2 Temperature Sensor Testing Circuit Readings……..18

ABSTRACT

Our project's main objective is to contribute to a human cause. Human detection in an unmanned area can be done efficiently by a robotic system. The task of identifying human being in rescue operations is difficult for the robotic agent but it is simple for the human agent. In order to detect a human body, an autonomous robot must be equipped with a specific set of sensors that provide information about the presence of a person in the environment around.

This alive human body detection system proposes two levels sensing system: First Level is the PIR sensors to detect the existence of the living human beings and Second Level is the distance sensor to measure the distance of the human body from the robotic vehicle. Temperature sensor has been used as a high temperature alert to prevent the robot from getting damaged. The robot is made a PC controlled vehicle by using wireless technology. The communication between personal computer and microcontroller is taking place by the means of ZigBee Transceiver.

As it is a wireless robot it can be easily mobilized and controlled. Hence, wireless communication increases the effectiveness of the detection process. Whenever any human motion is detected, an alarm is raised. This system has the potential to achieve high performance in detecting movement of humans in devastated environments relatively quickly and cost-effectively.

CHAPTER 1

Introduction

In this project, an efficient approach for detecting motion of humans in destructed environments using a pc controlled robot has been proposed. Natural calamities do occur and they are unstoppable producing a devastating effect and they see no difference between human beings and material. Hence a lot of times humans are buried among the debris and it becomes impossible to detect them. A timely rescue can only save the people who are buried and wounded. Detection by rescue workers becomes time consuming and due to the vast area that gets affected it becomes more difficult. So the project proposes a pc controlled robotic vehicle that moves in these affected areas and helps in identifying the alive people and thereby, contributing in the rescue operations.

Another application of our robotic vehicle lies in the detection of any intruder which has become very important in today's criminal environment which surrounds us at almost every place. Hence, designing such a detector can prevent various terrorist attacks and also can contribute in the home security system. It can also help in saving electricity which is a very costly resource nowadays. The robot consists of Arduino board which works as the central control unit of all the operations. PIR sensor, ultrasonic sensor and temperature sensor are interfaced with the Arduino. The PIR (Passive Infra-Red) Sensor is a pyroelectric device that detects motion by measuring changes in the infrared (heat) levels emitted by surrounding objects. When motion is detected, the

PIR sensor outputs a high signal on its output pin. This logic signal can be read by a microcontroller present in the Arduino. Ultrasonic sensor is used to measure the distance of the human body from the robotic vehicle and temperature sensor prevents the robot from entering in the areas of high temperature. Having detected the sign of alive human, the sensors trigger the microcontroller to switch on the buzzer or led. The ZigBee has been used to give direction to the robot so that it can move wirelessly. The distance and temperature measurements and PIR sensor output signals are sent via XBee router, which is interfaced with Arduino, to the XBee coordinator which is connected to the PC. Hence, this model effectively proposes wireless monitoring and governing of the robot.

Our project would be consisting of three modules:

- Control Room Module
- Power Supply Module
- Robot Module

1.1 Block Diagrams:

1.1.1 Control Room Module:

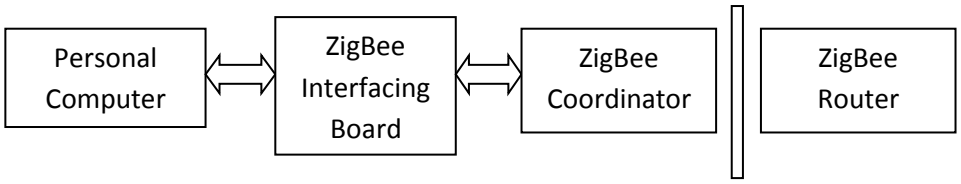

Figure: 1.1: Block Diagram of Control Room Module

1.1.2 Robot Module:

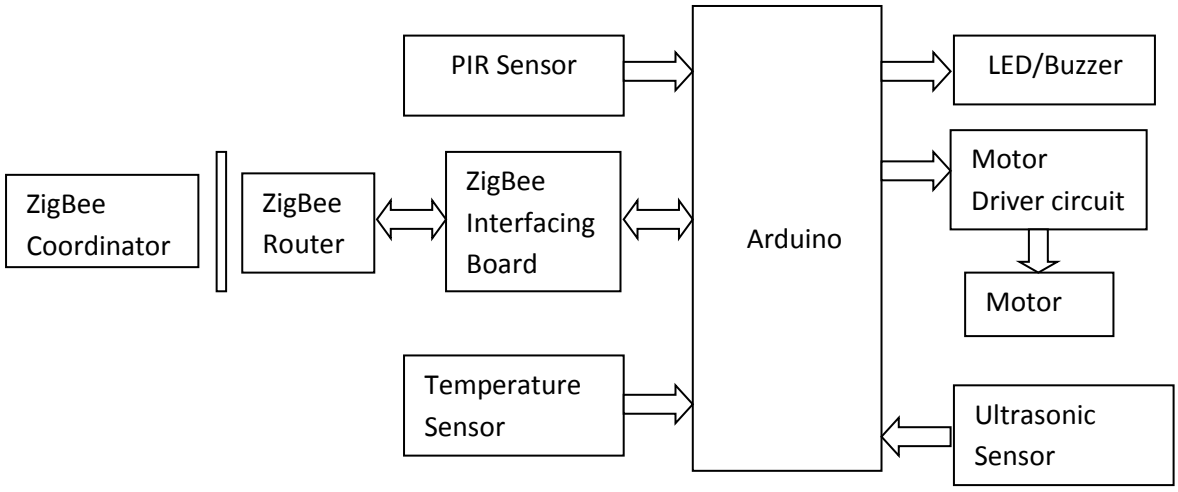

Figure 1.2: Block Diagram of Robot Module

CHAPTER 2

Technology Background

2.1 Arduino 1.6

The Arduino software is a suite for use with the Arduino Microcontrollers, which we use on our smaller robots. The tool is an IDE, or Integrated Development Environment. It contains a text editor, a compiler, and a serial communication terminal. The Arduino programming language is an implementation of Wiring, a similar physical computing platform, which is based on the Processing multimedia programming environment.

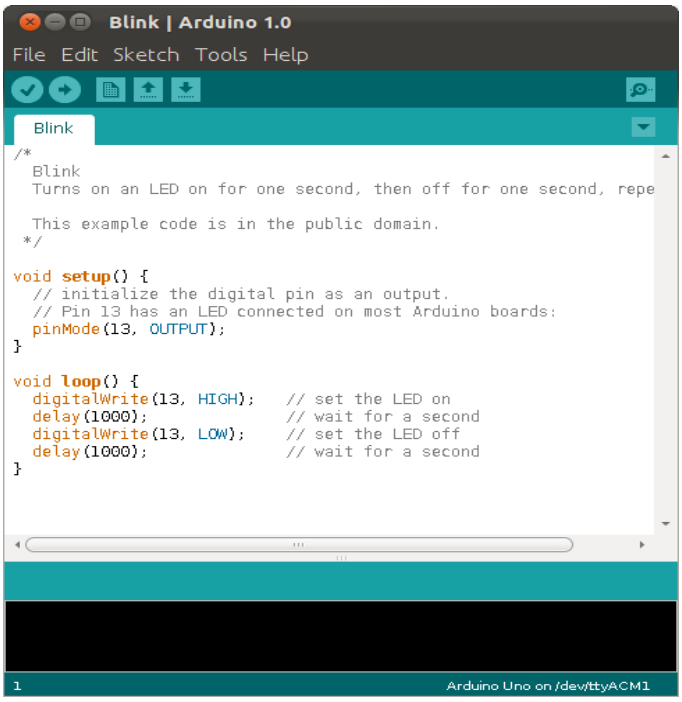

Figure 2.1: Arduino Software

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. It is designed to introduce programming to artists and other newcomers unfamiliar

with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with a single click. A program or code written for Arduino is called a "sketch".

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

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

2.2 X-CTU

XCTU is a free multi-platform application designed to enable developers to interact with Digi RF modules through a simple-to-use graphical interface. It includes new tools that make it easy to set-up, configure and test XBee RF modules. XCTU includes all of the tools a developer needs to quickly get up and running with XBee.

Unique features like graphical network view, which graphically represents the XBee network along with the signal strength of each connection, and the XBee API frame builder, which intuitively helps to build and interpret API frames for XBees being used in API mode, combine to make development on the XBee platform easier than ever.

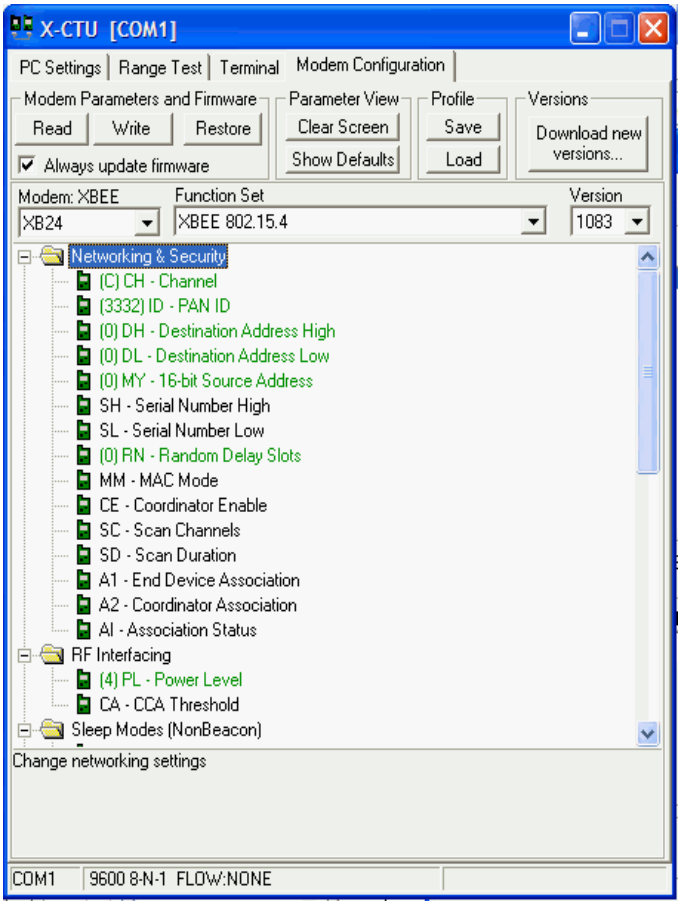

Figure 2.2: X-CTU Software

Major points about XCTU can be listed as follows:

- One can manage and configure multiple RF devices, even remotely (over-the-air) connected devices.

- The firmware update process seamlessly restores the module settings, automatically handling mode and baud rate changes.
- Two specific API and AT consoles, have been designed from scratch to communicate with the radio devices.
- One can now save the console sessions and load them in a different PC running XCTU.
- XCTU includes a set of embedded tools that can be executed without having any RF module connected.
- An update process allows you to automatically update the application itself and the radio firmware library without needing to download any extra files.
- XCTU contains complete and comprehensive documentation which can be accessed at any time.

CHAPTER 3

Proposed Hardware

3.1 Arduino UNO

The Arduino Uno is a microcontroller board based on the ATmega32-8. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started. The Atmega328 has 32KB of flash memory for storing code (of which 0.5KB is used for the boot loader). It has also 2KB of SRAM and 1KB of EEPROM (which can be read and

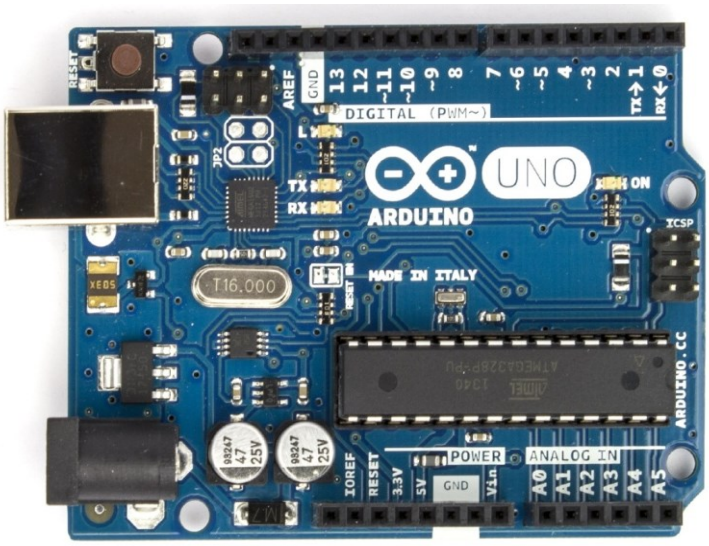

Figure 3.1: Arduino UNO

written with the EEPROM library). The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication.

3.1.1 Technical Specifications

- Microcontroller - ATmega328
- Operating Voltage - 5V
- Input Voltage (recommended) - 7V to 12V
- Input Voltage (limits) - 6V to 20V
- Digital I/O Pins - 14 (of which 6 provide PWM output)
- Analog Input Pins - 6
- DC Current per I/O Pin - 40mA
- DC Current for 3.3V Pin - 50mA
- Flash Memory - 32KB of which 0.5KB used by boot loader
- SRAM - 2KB
- EEPROM - 1KB
- Clock Speed - 16 MHz

3.1.2 Power Pins

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically .External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm centre-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector .The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN** - The input voltage to the Arduino board when it is using an external power source (as opposed to 5 volts from the USB connection or other regulated power source).

- **5V** - The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3.3V** - A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50mA.
- **GND** - Ground pins

3.1.3 Digital I/O Pins

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40mA and has an internal pull-up resistor (disconnected by default) of 20-50 kΩ. In addition, some pins have specialized functions:

- **Serial**: Pins 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts**: Pins 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- **PWM**: Pins 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function.
- **SPI**: Pins 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED**: Pin 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

3.1.4 Analog Input Pins

The Uno has 6 analog inputs, each of which provides 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **I2C**: Pins 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the Wire library. There are a couple of other pins on the board.
- **AREF**: Reference voltage for the analog inputs. Used with analogReference().
- **Reset**: Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

3.1.5 Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160mil (0.16").

3.1.6 USB over Current Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

3.1.7 How to use Arduino?

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

3.1.8 Functions used in Arduino

- **pinMode()**: Configures the specified pin to behave either as an input or an output.

  Syntax: pinMode(pin, mode)

  Parameters: Pin - the number of the pin whose mode you wish to set

- **digitalWrite()**: Write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW. If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pull-up on the input pin.

  Syntax: digitalWrite(pin, value)

  Parameter: Pin - the pin number

Value - HIGH or LOW

- **digitalRead()**: Reads the value from a specified digital pin, either HIGH or LOW.

  Syntax: digitalRead(pin)

  Parameters: Pin - the number of the digital pin you want to read (*int*).

- **analogReference()**:Configures the reference voltage used for analog input (i.e. the value used as the top of the input range). The default type corresponds to analog reference of 5 volts (on 5V Arduino boards) or 3.3 volts (on 3.3V Arduino boards) and external type corresponds to voltage applied to AREF pin (0 to 5V only).

  Syntax: analogReference(type)

  Parameters: type - which type of reference to use (DEFAULT, EXTERNAL).

- **analogRead()**: Reads the value from the specified analog pin. The Arduino board contains a 6 channel 10-bit analog to digital converter. This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023. This yields a resolution between readings of: 5 volts / 1024 units or, 0.0049 volts (4.9 mV) per unit. The input range and resolution can be changed using analogReference().It takes about 100 microseconds (0.0001 s) to read an analog input, so the maximum reading rate is about 10,000 times a second.

  Syntax: analogRead(pin)

  Parameters: Pin - the number of the analog input pin to read from

- **analogWrite()**: Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightness's or drive a motor at various speeds. After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite() (or a call to digitalRead() or digitalWrite() on the same pin). The frequency of the PWM signal on most pins is approximately 490 Hz. On the Uno and similar boards, pins 5 and 6 have a frequency of approximately 980 Hz. On most Arduino boards (those with the ATmega168 or ATmega328), this function works on pins 3, 5, 6, 9, 10, and 11.

  Syntax: analogWrite(pin, value)

  Parameters: Pin - the pin to write to

  Value - the duty cycle: between 0 and 255

- **Millis ()**: Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

- **Delay ()**: Pauses the program for the amount of time (in milliseconds) specified as parameter.

  Syntax: delay (ms)

Parameters: ms - the number of milliseconds to pause (unsigned long).

- **delayMicroseconds()**: Pauses the program for the amount of time (in microseconds) specified as parameter. There are a thousand microseconds in a millisecond, and a million microseconds in a second. Currently, the largest value that will produce an accurate delay is 16383. This could change in future Arduino releases. For delays longer than a few thousand microseconds, you should use delay () instead.

  Syntax: delayMicroseconds(us)

  Parameters: us - the number of microseconds to pause (unsigned int)

- **pulseIn**: Reads a pulse (either HIGH or LOW) on a pin.

  For example, if value is HIGH, pulseIn() waits for the pin to go HIGH, starts timing, then waits for the pin to go LOW and stops timing. This function returns the length of the pulse in microseconds. It gives up and returns 0 if no pulse starts within a specified time out. The timing of this function has been determined empirically and will probably show errors in longer pulses. It works on pulses from 10 microseconds to 3 minutes in length.

  Syntax: pulseIn(pin, value)

  pulseIn(pin, value, timeout)

  Parameters: pin - the number of the pin on which you want to read the pulse.

  Value - type of pulse to read: either HIGH or LOW.

  Timeout - the number of microseconds to wait for the pulse to start

- **print()**: Prints data to the serial port as human-readable ASCII text. This command can take many forms. Numbers are printed using an ASCII character for each digit. Floats are similarly printed as ASCII digits, defaulting to two decimal places. Bytes are sent as a single character. An optional second parameter specifies the base (format) to use; permitted values are BIN (binary, or base 2), OCT (octal, or base 8), DEC (decimal, or base 10), HEX (hexadecimal, or base 16). For floating point numbers, this parameter specifies the number of decimal places to use. For example: Serial.print(78) gives "78"

  Serial.print(78, BIN) gives "1001110"

- **write()**: Writes binary data to the serial port. This data is sent as a byte or series of bytes; to send the characters representing the digits of a number use the print() function instead. write() will return the number of bytes written, though reading that number is optional.

  Syntax: Serial.write(val)

  Serial.write(str)

  Serial.write(buf, len)

Parameters: val - a value to send as a single
str: a string to send as a series of bytes
buf: an array to send as a series of bytes
len: the length of the buffer

- **Begin()**: Sets the data rate in bits per second (baud) for serial data transmission. For communicating with the computer, one can use one of these rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200. One can, however, specify other rates - for An optional second argument configures the data, parity, and stop bits. The default is 8 data bits, no parity, one stop bit.

Syntax: Serial.begin(speed, config)
Serial.begin(speed)
Parameters: speed - in bits per second (baud)
config: sets data, parity, and stop bits.

3.2 ZigBee

ZigBee is a low-cost, low-power wireless mesh network standard targeted at wide development of long battery life devices in wireless control and monitoring applications. ZigBee is a specification for a suite of high-level communication protocols used to create personal area networks built from small, low-power digital radios. ZigBee is based on an IEEE 802.15.4 standard. Though its low power consumption limits transmission distances to 10–100 meters line-of-sight, ZigBee devices can transmit data over long distances by passing data through a mesh network of intermediate devices to reach more distant ones. ZigBee is typically used in low data rate applications that require long battery life and secure networking.

ZigBee has a defined rate of 250kbit/s, best suited for intermittent data transmissions from a sensor or input device. Applications include wireless light switches, electrical meters with in-home-displays, traffic management systems, and other consumer and industrial equipment that require short-range low-rate wireless data transfer.

The technology defined by the ZigBee specification is intended to be simpler and less expensive than other wireless personal area networks (WPANs), such as Bluetooth or Wi-Fi. The ZigBee standard operates on the IEEE 802.15.4 physical radio specification and operates in unlicensed bands including 2.4 GHz, 900 MHz and 868 MHz. ZigBee devices have low latency, which further reduces average current.

ZigBee devices are of three types:

- **ZigBee Coordinator (ZC):** The most capable device, the Coordinator forms the root of the network tree and might bridge to other networks. There is exactly one ZigBee Coordinator in each network since it is the device that started the network originally (the ZigBee LightLink specification also allows operation without a ZigBee Coordinator, making it more usable for over-the-shelf home products). It stores information about the network, including acting as the Trust Center & repository for security keys.
- **ZigBee Router (ZR):** Router can act as an intermediate router, passing on data from other devices.
- **ZigBee End Device (ZED):** Contains just enough functionality to talk to the parent node (either the Coordinator or a Router); it cannot relay data from other devices. This relationship allows the node to be asleep a significant amount of the time thereby giving long battery life. A ZED requires the least amount of memory, and therefore can be less expensive to manufacture than a ZR or ZC.

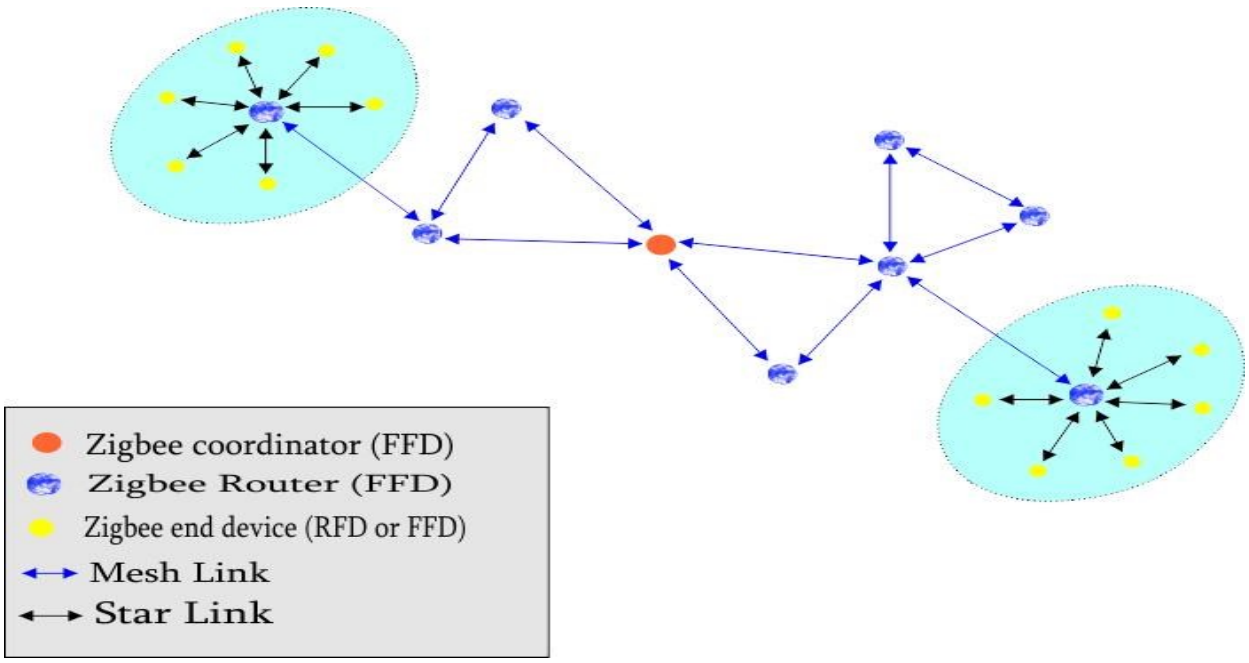

Figure 3.2: ZigBee Technology

The ZigBee network layer natively supports both star and tree networks, and generic Mesh networking. Every network must have one coordinator device, tasked with its creation, the control of its parameters and basic maintenance. Within star networks, the coordinator must be the central

node. Both trees and meshes allow the use of ZigBee routers to extend communication at the network level. ZigBee builds on the physical layer and media access control defined in IEEE standard 802.15.4 for low-rate WPANs. The specification includes four additional key components: network layer, application layer, ZigBee device objects (ZDOs) and manufacturer-defined application objects which allow for customization and favour total integration. ZDOs are responsible for a number of tasks, including keeping track of device roles, managing requests to join a network, as well as device discovery and security.

The current ZigBee protocols support beacon and non-beacon enabled networks. In non-beacon-enabled networks, an unslotted CSMA/CA channel access mechanism is used. In this type of network, ZigBee Routers typically have their receivers continuously active, requiring a more robust power supply. However, this allows for heterogeneous networks in which some devices receive continuously, while others only transmit when an external stimulus is detected. In beacon-enabled networks, the special network nodes called ZigBee Routers transmit periodic beacons to confirm their presence to other network nodes. Nodes may sleep between beacons, thus lowering their duty cycle and extending their battery life. Beacon intervals depend on data rate; they may range from 15.36 milliseconds to 251.65824 seconds at 250kbit/s, from 24 milliseconds to 393.216 seconds at 40kbit/s and from 48 milliseconds to 786.432 seconds at 20kbit/s. However, low duty cycle operation with long beacon intervals requires precise timing, which can conflict with the need for low product cost.

3.2.1 XBee

The XBee RF Modem from Digi International is a wireless transceiver. The XBee uses a fully implemented protocol for data communications which provides features needed for robust network communications in a wireless sensor network (WSN). Features such as addressing, acknowledgements and retries help ensure safe delivery of data to the intended node.

Figure 3.3: XBee S2

The XBee also has additional features beyond data communications for use in monitoring and control of remote devices. The XBee module comes in several versions but all have similar pinouts The XBee is a 20-pin DIP module with pin spacing of 2 mm (0.079 in) as opposed to typical pin spacing of 2.54 mm (0.1 in).

The XBee utilizes the IEEE 802.15.4 protocol which implements the following features:

- **Media Access**: A means to ensure two network nodes do not transmit at the same time causing data collisions and errors in transmission. XBee has a feature called Clear Channel Assessment in which before transmitting, an XBee node listens to see if the selected frequency channel is busy.
- **Addressing**: A means to ensure only the intended node uses the received data, allowing data to be sent from one point to another point. Or, point to multi-point by sending a broadcast meant for all nodes on the network. The XBee has two addressing options: a fixed 64-bit serial number (MAC address) which cannot be changed, and a 16-bit assignable address that allows over 64,000 addresses on a network.
- **Error Detection**: A means to verify data received at the node correctly. The XBee uses a checksum to help ensure received data contains no errors.
- **Acknowledgements & Retries**: A means to inform the transmitting node that the data was delivered successfully. Lacking this, at most 3 retries are performed in an effort to deliver the data.

3.2.2 Communication Modes

The XBee supports both an AT and an API (Application Programming Interface) mode for sending and receiving data at the controller. Both have their advantages.

- **AT Mode**: In AT Mode, also called Transparent Mode, just the message data itself is sent to the module and received by the controller. The protocol link between the two is transparent to the end user and it appears to be a nearly direct serial link between the nodes. This mode allows simple transmission and reception of serial data. AT Commands are used to configure the XBee. Even though the transmission and reception is the raw data, the message itself is passed between nodes encapsulated with needed information such as addressing and error checking bytes. This mode is primarily used in instances where an existing protocol cannot tolerate changes to the data format.
- **API Mode**: In API Mode, the programmer packages the data with needed information, such as destination address, type of packet, and checksum value. Also, the receiving node accepts the data with information such as source address, type of packet, signal strength, and checksum value. The advantages are the user can build a packet that includes important data, such as

destination address, and that the receiving node can pull from the packet information such as source address of the data. While more programming intensive, API Mode allows the user greater flexibility and increased reliability in some cases.

Both sides do not need to be in the same mode. Data may be sent in API Mode and received in AT Mode or vice-versa. The mode defines the communications link between the PC or controller and the XBee modem, and not between XBee modules.

3.2.3 XBee Configuration

The XBee has multitude of configuration settings to deal with addressing, interfacing and input and output control. X-CTU interface allows range testing with loopback, modem configuration and a terminal window for communications, testing and configuration. Here is a list of some common configuration settings of XBee.

Group	**Command**	**Meaning and Use**
Networking & Security	CH	Channel: Sets the operating frequency channel within the 2.4 GHZ band. This may be modified to find a clearer channel to separate XBee networks.
	ID	PAN ID: Essentially, the network ID. Different groups of XBee networks can be separated by setting up different PANs (Personal Area Networks). Default value is 3332.
	DL	Destination Low Address: The destination address of the XBee for which the transmitted packet is indented. We will use this often to define which node receives data. A hexadecimal value of FFFF performs a broadcast and sends data to all nodes on the PAN. The default value is 0.
	MY	Source Address: Sets the address of the node itself. This will be used often in all our configurations. The default value is 0.
	NI	Node Identifier: Sets a node's noun name.
Sleep Modes	SM	Sleep Mode: Allows the sleep mode to be selected for low power consumption (<10 μA).

Serial Interfacing	BD	Interface Data Rate: Sets baud rate of the serial data between the XBee and the controller or PC.
	AP	API Enable: Switches the XBee from transparent mode to a framed data version where the data must be manually framed with other information
	RO	Packetization Timeout: In building a packet to be transmitted, this sets the length of time the XBee waits for another character before the packet is sent.
I/O Settings	D0-D8	Sets the function of the I/O pins on the XBee, such as digital output, input, ADC, RTS, CTS and other for their proper functioning.
	P0,P1	Configures the function of PWM0 and PWM1
	RP	Sets the time length for the RSSI output.
	M0,M1	Sets the PWM value of the PWM outputs
	IR	Sample Rate: The XBee can be configured to automatically send data from digital I/O or ADC's. It requires the receiving node to be in API Mode and the data parsed for the I/O values.
Diagnostics	DB	Received Signal Strength: The XBee can be polled to send back the RSSI level of the last packet received.
	EC	CCA Failures: The protocol performs clear channel assessment (CCA); that is, it listens to the RF levels before it transmits. If it cannot get an opening, the packet will fail and the CCA counter will be incremented.
	EA	ACK Failures: If a packet is transmitted but receives no acknowledgement that data reached the destination, EA is incremented. The XBee performs 2 retries before failure. Additional retries can be added by using RR setting.
AT Command Options	CT	AT Command Timeout: Once in Command Mode, sets the length of the delay before the XBee returns to normal operation.

	GT	Guard Time: When switching into AT Command Mode, defines how long the guard times should be (absence of data before the command line) so that accidental mode change is not performed.
Configuration	WR	Write the configuration to non-volatile memory.
	RE	Restore to default configuration in volatile memory.

Table 3.1: XBee Configuration Settings

3.2.4 XBee - Series 2 (ZigBee Mesh)

XBee XB24-Z7WIT-004 module from Digi Series has improvement in terms of the power output and data protocol. Series 2 modules allow one to create complex mesh networks based on the XBee ZB ZigBee mesh firmware. These modules allow a very reliable and simple communication between microcontrollers, computers, systems, etc. with a serial port. Point to point and multi-point networks are supported. These are essentially the same hardware as the older Series 2.5, but have updated firmware. They will work with Series 2.5 modules if the firmware is updated through X-CTU.

3.2.5 XBee Pin Description

Important pins can be described as follows:

- **DOUT and DIN**: These are the pins through which serial data is received by our controller or PC (DOUT) and sent to the XBee (Din). This data may be either for transmission between XBee modules or for setting and reading configuration information of the XBee. The default data rate is 9600 baud (bps) using asynchronous serial communications.
- **RESET**: A momentary low on this pin will reset the XBee to the saved configuration settings.
- **CTS/RTS/DTR**: These are used for handshaking between the XBee and the controller or the PC. The

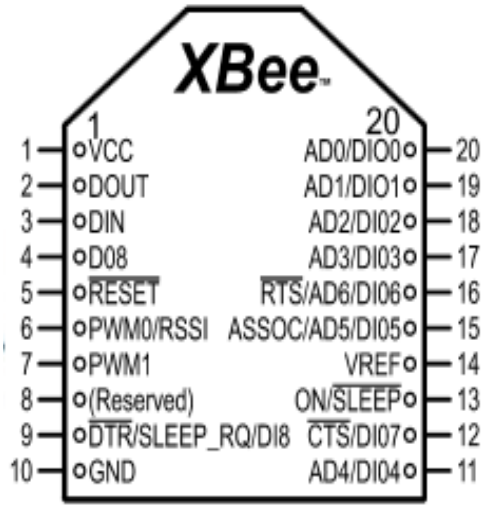

Figure 3.4: XBee Pin Configuration

XBee will not send data out through the DOUT line to the controller unless the RTS line is held low.

This allows the controller to signal to the XBee that it is ready to receive more data. DTR is typically used by the XBee when downloading new firmware, and therefore firmware updates can only be done using XBee adapter boards such as the Parallax USB Adapter Board that implement this connection. When transmitting, the XBee can signal to the controller through the CTS line that it is ready to send more data. CTS is seldom needed because the XBee sends data out by radio much more quickly than it accepts data from the controller.

- **DIO0–DIO7/D08**: These are used as standard 3.3 V digital inputs and outputs. The XBee can be controlled to set the state of the pins. They can also be used in "line passing" so that the state of a pin on one XBee (high or low) is reflected on the corresponding pin of another XBee.

- **AD0 to AD6**: These are 10-bit Analog to Digital Converter (ADC) inputs to the XBee. While we cannot directly read these values, some can also be used in "line passing" so that the amount of voltage on a pin on one XBee is reflected by the amount of voltage (PWM) on the corresponding pin of another XBee.

- **RSSI**: The XBee can report the strength of the received RF signal as PWM output on this pin. This value can also be retrieved using AT commands or as part of a packet in API Mode.

- **PWM0/1**: These pins can be set for 10-bit pulse width modulated output, which can be used directly or filtered for analog output. They can also be controlled using “line passing” by the analog input on another XBee.

- **ASSOC**: When configured, the XBee can be set to join an existing network and assigned certain parameters. In this tutorial we will manually configure the XBee in the network instead of joining networks.

3.2.6 XBee (Series 2) ZB Features

- Supply Voltage- 2.1 - 3.6VDC
- Transmit Current- 35mA / 45mA @ 3.3VDC
- Receive Current- 38mA / 40mA @ 3.3VDC
- RF data rate- 250 kbps
- 2mW output (+3dBm)
- 400ft (120m) range
- 40m Indoor/urban range

- Built-in antenna
- 6 10-bit ADC input pins
- 8 digital IO pins
- 128-bit encryption
- Local or over-air configuration
- AT or API command set
- Operating Temperature- (-40º C to +85º C)
- Frequency Band- 2.4GHz
- Receiver Sensitivity- (-96dBm)
- Serial Data Rate- (1200bps – 1Mbps)

3.3 XBee USB Adaptor Board

This unit works with all XBee modules including the Series 1 and Series 2, standard and Pro version. It also supports Wi-Fi Bee & Bluetooth Bee. Plug the unit into the XBee Explorer, attach a mini USB cable, and one can have direct access to the serial and programming pins on the XBee unit. This adapter board provides an easy interface to the XBee or XBee Pro modules by converting the 2mm pin spacing to breadboard friendly 0.100" spacing. The adapter boards also enable a means to connect pluggable wires or solder connections and also provide mounting holes.

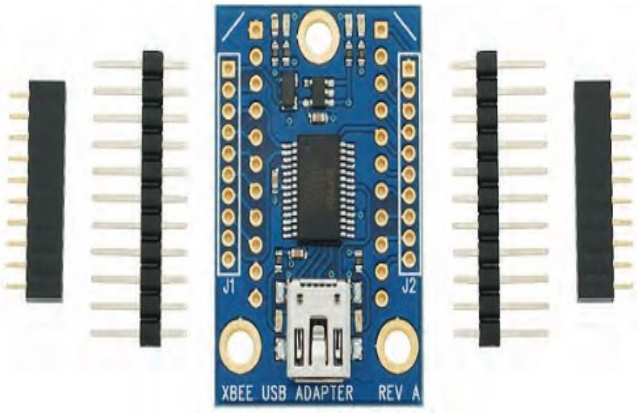

Figure 3.5: XBee USB Adaptor Board

This board doesn't come assembled. They provide an easy interface for configuring an XBee Module via a PC. We can also use this module to change configuration of the XBee devices using PC via USB port using XCTU software. There's also a reset button, and a voltage regulator to supply the XBee with plenty of power. They are also equipped with 4 status indicator LEDs which show power, status of the data being transmitted or received, signal strength of the data reception and the association. The function corresponding to each LED can be listed as follows:

- **Green**: ON/Not Sleeping. This will typically be on unless you configure the XBee for one of the low- power sleep modes.
- **Yellow**: Power. Indicates 3.3 V power is available.

- **Red**: Associate. This LED will typically be blinking unless using the associate mode of joining a network or by changing the pin configuration.
- **Blue**: RSSI indicator. The XBee PWM output indicated strength of received RF signal. This LED will light for several seconds anytime the XBee receives RF data addressed to it. A slight dimming at low power levels may be noticeable.
- **Red**: USB Transmit
- **Green**: USB Receive

3.4 Arduino XBee Shield

The XBee Shield gives the Arduino a seamless interface to XBee – one of the most popular wireless platforms around. With XBee, instead of being tied down by a serial cable – inches away from a paired device – the Arduino can pass data over the air to another device hundreds of feet away. The Arduino XBee shield can be used with different XBee modules.

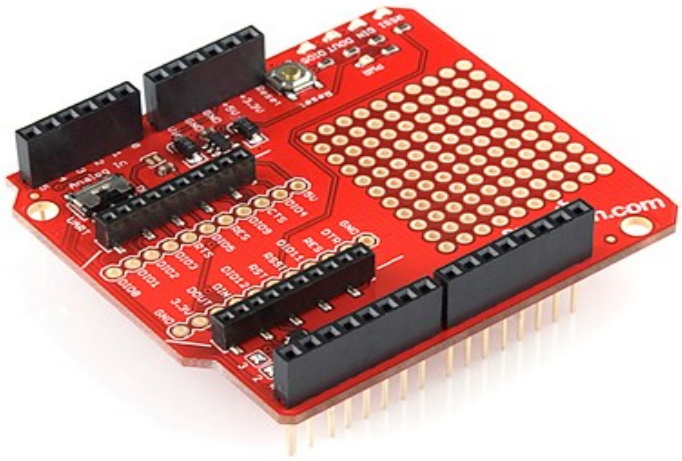

Figure 3.6: Arduino XBee Shield

The Xbee shield has two jumpers (the small removable plastic sleeves that each fit onto two of the three pins labelled Xbee/USB). These determine how the Xbee's serial communication connects to the serial communication between the microcontroller (ATmega8 or ATmega168) and FTDI USB-to-serial chip on the Arduino board.

With the jumpers in the Xbee position (i.e. on the two pins towards the interior of the board), the DOUT pin of the Xbee module is connected to the RX pin of the microcontroller; and DIN is connected to TX. The RX and TX pins of the microcontroller are still connected to the TX and RX pins (respectively) of the FTDI chip - data sent from the microcontroller will be transmitted to the computer via USB as well as being sent wirelessly by the Xbee module. The microcontroller, however, will only be able to receive data from the Xbee module, not over USB from the computer.

With the jumpers in the USB position (i.e. on the two pins nearest the edge of the board), the DOUT pin the Xbee module is connected to the RX pin of the FTDI chip, and DIN on the Xbee module is connected to the TX pin of the FTDI chip. This means that the Xbee module can communicate directly with the computer - however, this only works if the microcontroller has been

removed from the Arduino board. If the microcontroller is left in the Arduino board, it will be able to talk to the computer normally via USB, but neither the computer nor the microcontroller will be able to talk to the Xbee module.

3.5 Temperature Sensor

An analog temperature sensor is a chip that tells what the ambient temperature is. These sensors use a solid-state technique to determine the temperature. They don't use mercury (like old thermometers), bimetallic strips (like in some home thermo-meters or stoves), nor do they use thermistors (temperature sensitive resistors). Instead, they use the fact as temperature increases, the voltage across a diode increases at a known rate. Technically, this is actually the voltage drop between the base and emitter - the V_{be} - of a transistor. By precisely amplifying the voltage change, it is easy to generate an analog signal that is directly proportional to temperature.

Figure 3.7: Temperature Sensor

The semiconductor temperature sensors offer high accuracy and high linearity over an operating range of about –55ºC to +150ºC. Internal amplifiers can scale the output to convenient values, such as 10mV/ºC. Since these sensors have no moving parts, they are precise, never wear out, don't need calibration, work under many environmental conditions, and are consistent between sensors and readings. Moreover they are very inexpensive and quite easy to use.
In our project, we would be using LM35DZ temperature sensor.

3.5.1 LM35

The LM35 series are precision integrated-circuit temperature sensors, with an output voltage linearly proportional to the Centigrade temperature.
Advantages of LM35 can be listed as follows:

- LM35 has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant

Figure 3.8: LM35

voltage from the output to obtain convenient Centigrade scaling.

- It do not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to +150°C temperature range.
- Low cost is assured by trimming and calibration at the wafer level.
- The low output impedance, linear output, and precise inherent calibration of the LM35 make interfacing to readout or control circuitry especially easy.
- The device is used with single power supplies, or with plus and minus supplies.
- As the LM35 draws only 60μA from the supply, it has very low self-heating of less than 0.1°C in still air.
- The LM35 is rated to operate over a −55°C to +150°C temperature range.

The general equation used to convert output voltage to temperature is:

Temperature (^oC) = Vout * ($100^oC/V$)

So if output voltage is 1V, then, Temperature = 100^oC. Thus, this shows that output voltage varies linearly with temperature.

Absolute Maximum Ratings are as follows:

- Supply Voltage = +35V to -0.2V
- Output Voltage = +6V to -1.0V
- Output Current = 10mA
- Storage Temperature = -60^0C to $+180^0C$

3.6 PIR Sensor

A passive infrared sensor (PIR sensor) is an electronic sensor that measures infrared (IR) light radiating from objects in its field of view. They are most often used in PIR-based motion detectors. All objects with a temperature above absolute zero emit heat energy in the form of radiation. Usually this radiation is invisible to the human eye because it radiates at infrared wavelengths, but it can be detected by electronic devices designed for such a purpose. A PIR-based motion detector is used to sense movement of people, animals, or other objects. They are commonly used in burglar alarms and automatically-activated lighting systems.

Figure 3.9: PIR Sensor

The term passive in this instance refers to the fact that PIR devices do not generate or radiate any energy for detection purposes. They work entirely by detecting the energy given off by other objects. PIR sensors don't detect or measure "heat"; instead they detect the infrared radiation emitted or reflected from an object.

Infrared radiation enters through the front of sensor, known as the 'sensor face'. At the core of a PIR sensor is a solid state sensor or set of sensors, made from pyroelectric materials—materials which generate energy when exposed to heat. Typically, the sensors are approximately 1/4 inch square (40 mm^2), and take the form of a thin film. Materials commonly used in PIR sensors include gallium nitride (GaN), caesium nitrate ($CsNO_3$).

Along with the pyroelectric sensor is a bunch of supporting circuitry, resistors and capacitors. It includes 3V DC regulator, 3-5V DC power, protection diode, PIR chip, adjustments for delay time and adjustments of sensitivity.

3.6.1 Basic Principle

An individual PIR sensor detects changes in the amount of infrared radiation impinging upon it, which varies depending on the temperature and surface characteristics of the objects in front of the sensor. When an object, such as a human, passes in front of the background, such as a wall, the temperature at that point in the sensor's field of view will rise from room temperature to body temperature, and then back again. The sensor converts the resulting change in the incoming infrared radiation into a change in the output voltage. Moving objects of similar temperature to the background but different surface characteristics may also have a different infrared emission pattern, and thus sometimes trigger the detector.

3.6.2 Differential Detection

Pairs of sensor elements may be wired as opposite inputs to a differential amplifier. In such a configuration, the PIR measurements cancel each other so that the average temperature of the field of view is removed from the electrical signal; an increase of IR energy across the entire sensor is self-cancelling and will not trigger the device.

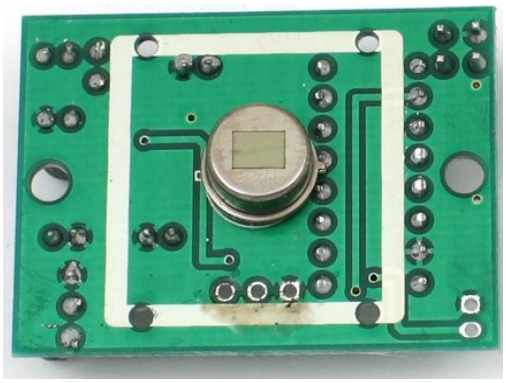

Figure 3.10: Differential Detection

This allows the device to resist false indications of change in the event of being exposed to brief flashes of light

or field-wide illumination. At the same time, this differential arrangement minimizes common-mode interference, allowing the device to resist triggering due to nearby electric fields. However, a differential pair of sensors cannot measure temperature in this configuration, and therefore is only useful for motion detection.

3.6.3 How does PIR Sensor work?

The PIR sensor itself has two slots in it; each slot is made of a special material that is sensitive to IR. The lens used here is not really doing much. The two slots can 'see' out past some distance (basically the sensitivity of the sensor). When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors. When a warm body like human or animal passes within the field of detection, it intercepts one half of the PIR sensor.

This causes a positive differential change between the two halves. When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected

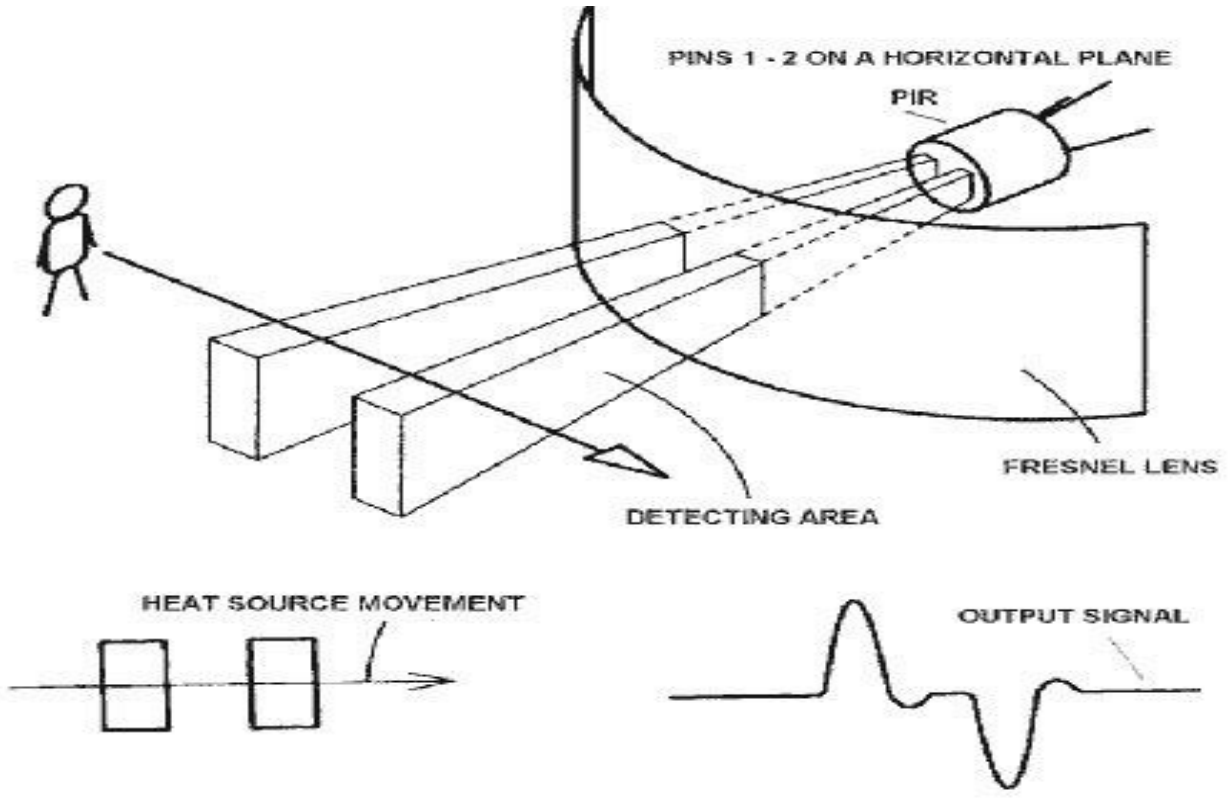

Figure 3.11: Working of PIR Sensor

The IR sensor itself is housed in a hermetically sealed metal can to improve noise/temperature/humidity immunity. There is a window made of IR - transmissive material

(typically coated silicon since that is very easy to come by) that protects the sensing element. Behind the window are the two balanced sensors. In order to have a detection area that is much larger, we use a simple lens such as those found in a camera: they condenses a large area (such as a landscape) into a small one (on film or a CCD sensor). For this reason the sensors are actually Fresnel lenses. In order to get a scattering of multiple small areas, lens is splitted into multiple sections, each section of which is a Fresnel lens.

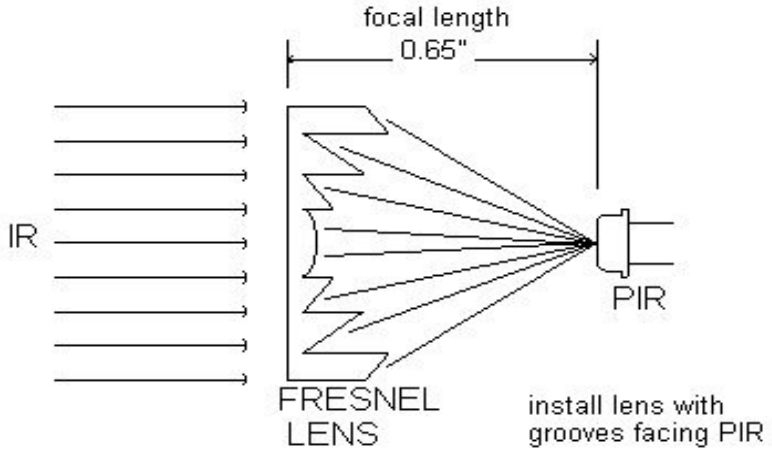

Figure 3.12: Fresnel Lens

The pyroelectric sensor is made of a crystalline material that generates a surface electric charge when exposed to heat in the form of infrared radiation. When the amount of radiation striking the crystal changes, the amount of charge also changes and can then be measured with a sensitive FET device built into the sensor.

The sensor elements are sensitive to radiation over a wide range so a filter window is added to the TO5 package to limit detectable radiation to the 8 to 14mm range which is most sensitive to human body radiation.

There are two modes in the PIR sensor: Retriggering and non-retriggering. Let an LED is connected to the output pin of the PIR sensor so that it can be used for motion detection. If the LED does not stay on when moving in front of it but actually turns on and off every second or so, then this is called "non-retriggering". This takes place when the jumper is placed in the L position. If the LED stays on the entire time that something is moving, then this is called "retriggering". This can be used by placing the jumper in H position.

The B1SS001 chip takes the output of the sensor and does some minor processing on it to emit a digital output pulse from the analog sensor.

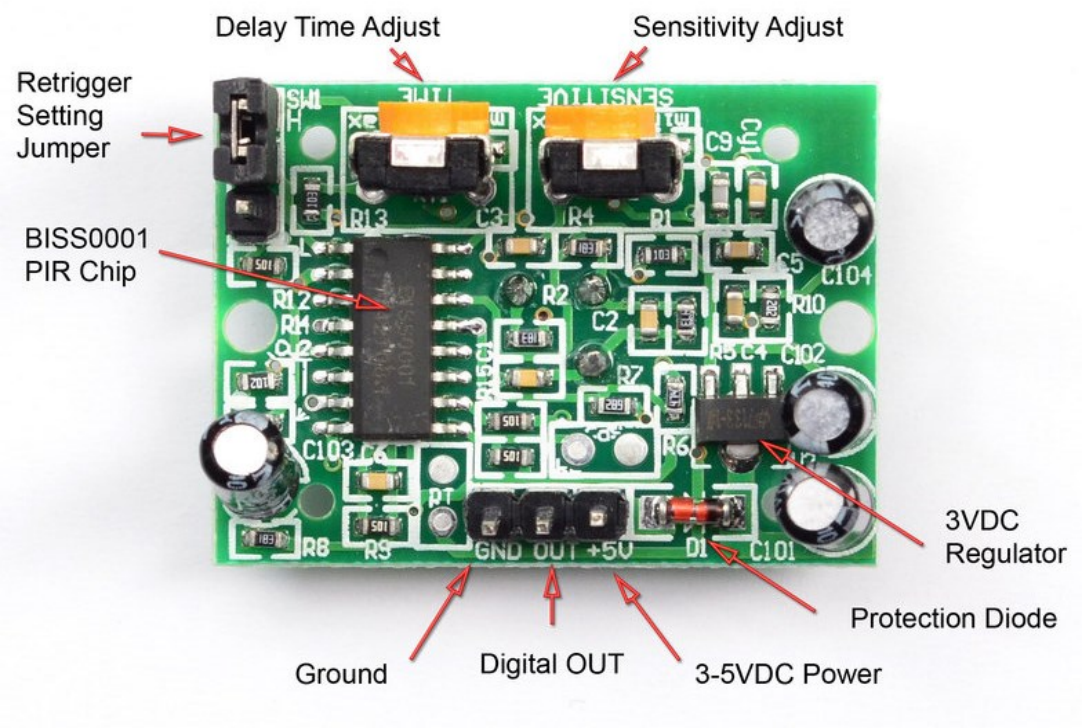

Figure 3.13: Overview of PIR Sensor

3.6.4 Changing Pulse Time and Timeout Length

There are two 'timeouts' associated with the PIR sensor. One is the "Tx" timeout: how long the LED is lit after it detects movement. The second is the "Ti" timeout which is how long the LED is guaranteed to be off when there is no movement. These are not easily changed but it can be done by using soldering techniques.

Formulae used for calculating R10, C6, R9, and C7 corresponding to a particular Tx and Ti:

$$Tx = 24576 * R10 * C6$$

$$Ti = 24 * R9 * C7$$

3.6.5 Technical Specifications

- **Sensitivity Range**: Its sensitivity range is up to 20 feet (6 metres) 110 degrees x 70 degrees detection range.
- **Power Supply**: 3.3V – 5V input voltage
- **Output**: Digital pulse high (3V) when triggered (motion detected) digital low when idle (no motion detected). Pulse lengths are determined by resistors and capacitors on the PCB and differ from sensor to sensor.

3.7 Ultrasonic Sensor

Figure 3.14: Ultrasonic Sensor

Ultrasonic transducers are transducers that convert ultrasound waves to electrical signals or vice versa. Those that both transmit and receive may also be called ultrasound transceivers; many ultrasound sensors besides being sensors are indeed transceivers because they can both sense and transmit. These devices work on a principle similar to that of transducers used in radar and sonar systems, which evaluate attributes of a target by interpreting the echoes from radio or sound waves, respectively. Active ultrasonic sensors generate high frequency sound waves and evaluate the echo which is received back by the sensor, measuring the time interval between sending the signal and receiving the echo to determine the distance to an object. Passive ultrasonic sensors are basically microphones that detect ultrasonic noise that is present under certain conditions, convert it to an electrical signal, and report it to a computer.

Piezoelectric crystals have the property of changing size when a voltage is applied; applying an alternating current (AC) across them causes them to oscillate at very high frequencies, thus producing very high frequency sound waves. These sound waves are then used for certain applications.

3.7.1 Measurement Principle of Ultrasonic Sensor

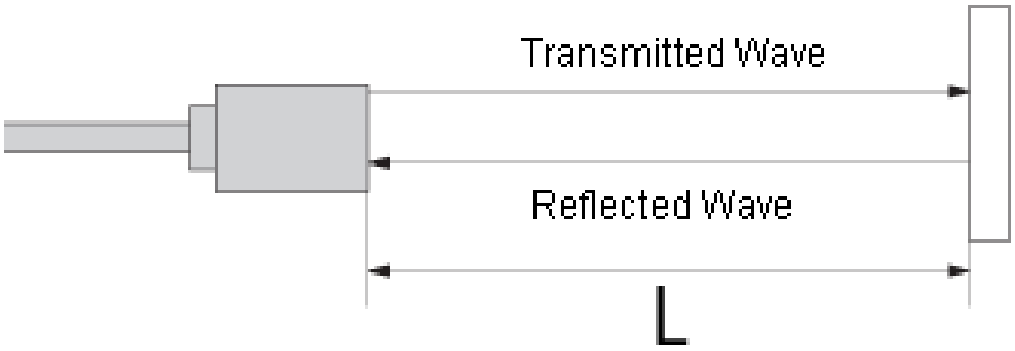

Figure 3.15: Principle of Ultrasonic Sensor

Ultrasonic sensor transmits ultrasonic waves from its sensor head and again receives the ultrasonic waves reflected from an object. By measuring the length of time from the transmission to reception of the sonic wave, it detects the position of the object.

The speed of sound "C" in air is C ≈ 331.5 + 0.61 θ (m/s), where θ is the air temperature (°C). The speed of sound changes as the air temperature changes which results in temperature-based distance measurement error. Ultrasound waves move straight forward in a uniform medium, and are reflected and transmitted at the boundary between differing media. This phenomenon is affected by the type and shape of the media. A human body in air causes considerable reflection and can be easily detected. The corresponding formula which can be used for distance measurement is as follows:

Distance = (Echo pulse width high time * Sound Velocity (340m/s)/2)

Distance in cm = (Echo pulse width high time (in us)*0.017)

3.7.2 HC-SR04

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules include ultrasonic transmitters, receiver and control circuit.

Figure 3.16: HC-SR04

Key points which have to be taken into consideration are as follows:

- The module is not suggested to connect directly to electric ports. If connected to electricity, the GND terminal should be connected in the module first, otherwise it will affect the normal work of the module.
- When tested objects, the range of area should not be less than 0.5 square meters and the plane requests as smooth as possible, otherwise, it will affect the results of measuring.

The basic principle of work can be described as follows:

- Using IO trigger for at least 10us high level signal
- The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
- If the signal is back, through high level, time of high output IO duration is the time from sending ultrasonic to returning.

This IO triggering triggers the transducer and ultrasound sound waves of frequency 40 KHz are produced which return as an echo after striking any object. The time it takes for obtaining the echo is used in distance calculation.

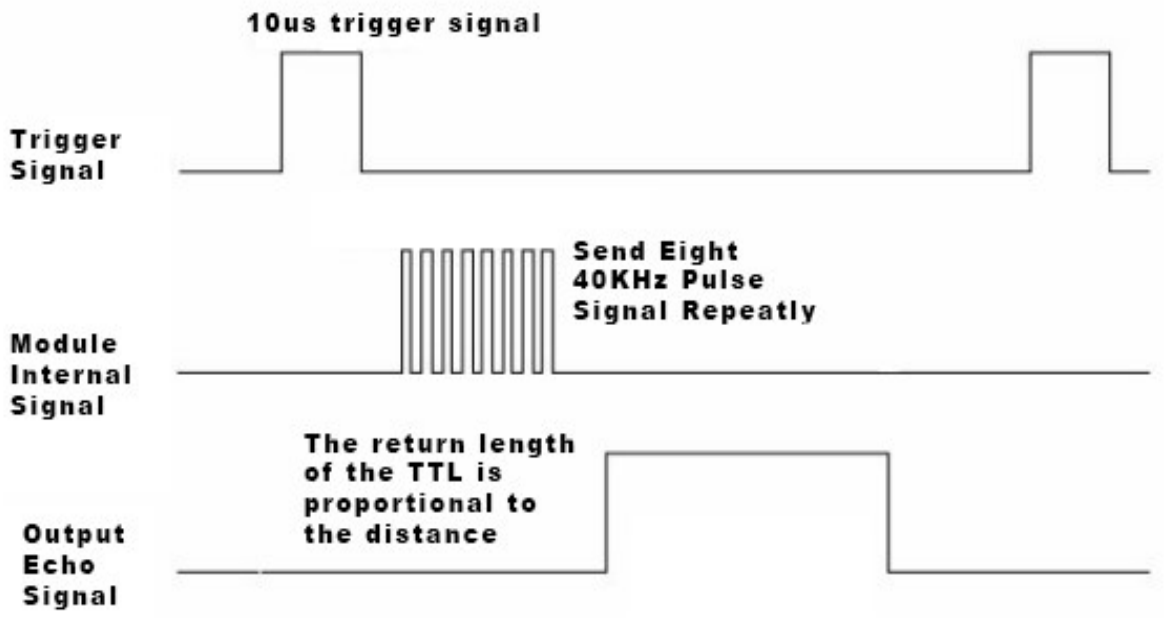

Figure 3.17: Timing Diagram showing working of HC-SR04

3.7.3 Electrical Specifications of HC-SR04

- **Working Voltage**: DC 5V
- **Working Current**: 15mA
- **Working Frequency**: 40Hz
- **Max Range**: 4m
- **Min Range**: 2cm
- **Measuring angle**: 15^0
- **Trigger Input Signal**: 10us TTL pulse
- **Echo Output Signal**: Input TTL lever signal and the range in proportion

3.8 DC Motors

A DC motor is any of a class of electrical machines that converts direct current electrical power into mechanical power. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic; to periodically change the direction of current flow in part of the motor. DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems.

A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances.

A geared DC Motor has a gear assembly attached to the motor. The speed of motor is counted in terms of rotations of the shaft per minute and is termed as RPM. The gear assembly helps in increasing the torque and reducing the speed. Using the correct combination of gears in a gear motor, its speed can be reduced to any desirable figure. This concept where gears reduce the speed of the vehicle but increase its torque is known as gear reduction.

Figure 3.18: DC Gear Motor

The DC motor works over a fair range of voltage. The higher the input voltage more is the RPM (rotations per minute) of the motor. For example, if the motor works in the range of 6-12V, it will have the least RPM at 6V and maximum at 12V. The gear having smaller radius will cover more RPM than the one with larger radius. However, the larger gear will give more torque to the smaller gear than vice versa.

The comparison of angular velocity between input gear (the one that transfers energy) to output gear gives the gear ratio. When multiple gears are connected together, conservation of energy is also followed. The direction in which the other gear rotates is always the opposite of the gear adjacent to it. In any DC motor, RPM and torque are inversely proportional. Hence the gear having more torque will provide a lesser RPM and converse. In a geared DC motor, the concept of pulse width modulation is applied.

3.8.1 General Specifications

- Supply Voltage: 4V to 12V
- Speed: 100rpm at 12V
- 6mm shaft diameter
- 22mm shaft length
- 143gms weight
- Stall torque: 10Kg-cm at stall current of 1.3Amp.
- Any wheel with 6mm diameter bore can be used with this motor

3.8.2 Transistor Circuit for Interfacing Motor with Arduino

The small DC motor is likely to use more power than an Arduino digital output can handle directly. If the motor is connected straight to an Arduino pin, there is a good chance that it could damage the Arduino.

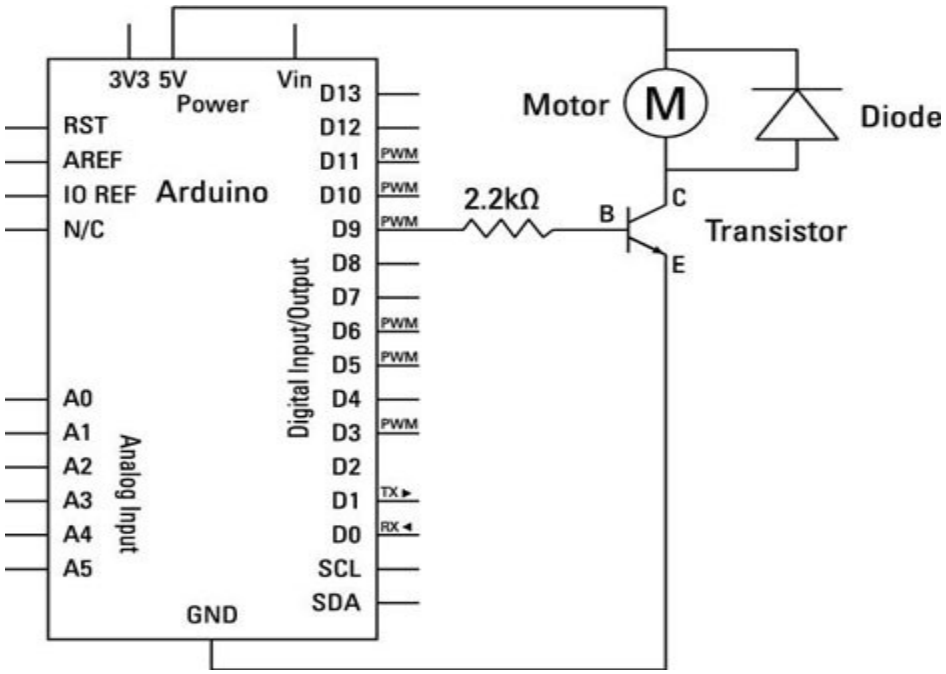

Figure 3.19: Transistor Circuit for Motor Interfacing

To power the motor, we need to send 5V through it and then on to ground. This voltage spins the motor, but we should have control of it. To give the Arduino control of the motor's power, and therefore its rotation, we place a transistor just after the motor. The transistor is an electrically operated switch that can be activated by your Arduino's digital pins. The transistor allows you the turn the motor circuit on and off.

This circuit works, but it still allows the chance of creating a reverse current because of the momentum of the motor as it slows down, or because the motor could be turned. If reverse current is generated, it travels from the negative side of the motor and tries to find the easiest route to ground. This route may be through the transistor or through the Arduino. We can't know for sure what will happen, so we need to provide a way to control this excess current. To be safe, we place a diode across the motor. The diode faces toward the source of the voltage, meaning that the voltage is forced through the motor, which is what we want. If current is generated in the opposite direction, it is can now be blocked from flowing into the Arduino. If the diode is placed in the wrong way, the

current bypasses the motor and we create a short circuit. The short circuit tries to ground all the available current and could break the USB port or at the very least, show a warning message, informing that the USB port is drawing too much power.

In this project, we have used PN2222 transistor, 330ohms resistor and 1N4001 diode.

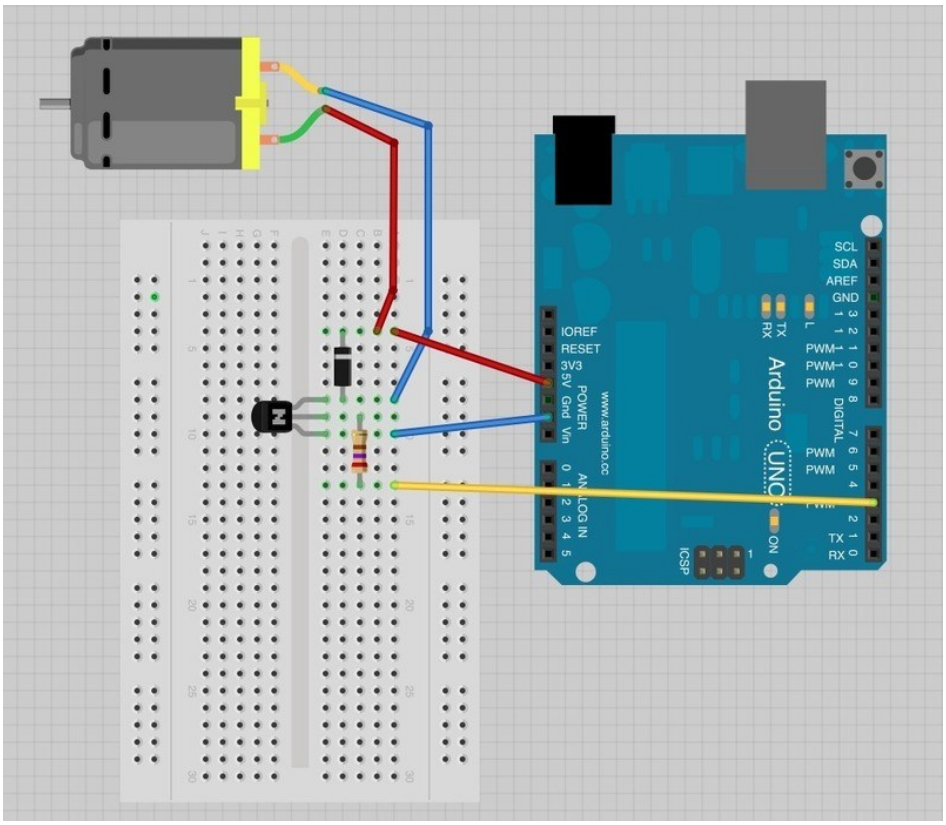

Figure 3.20: Breadboard Layout of Motor Interfacing Circuit

CHAPTER 4

Methodology

4.1 Module 1: Mechanical Construction of Robot

Robot has been built using following steps:

- A high quality metal chassis is used.
  - Length = 19.5 cm (7.5 inches) (motor and wheels are not included)
  - Breadth = 10.0 cm (4.2 inches) (motor and wheels are not included)
- DC gear motors are mounted on the chassis. Unscrew the nut of motor and insert the motor
- Fix wheels on the motor shaft. Wheel for the gear motor is having following specifications and these can be associated with 6mm diameter shaft.
  - Diameter = 7.2 cm
  - Thickness = 2 cm
  - Colour = Black
  - Tyre = Tracked rubber

Figure 4.1: Fixing DC Motors and Wheels on the metal chassis

4.2 Module 2: XBee Configuration Settings

Two XBee ZB24-B modules are configured by using X-CTU software. The module is firstly interfaced with XBee USB adaptor board and then it is connected to the personal computer via USB cable. Start X-CTU software and following steps are followed.

4.2.1 PC Settings

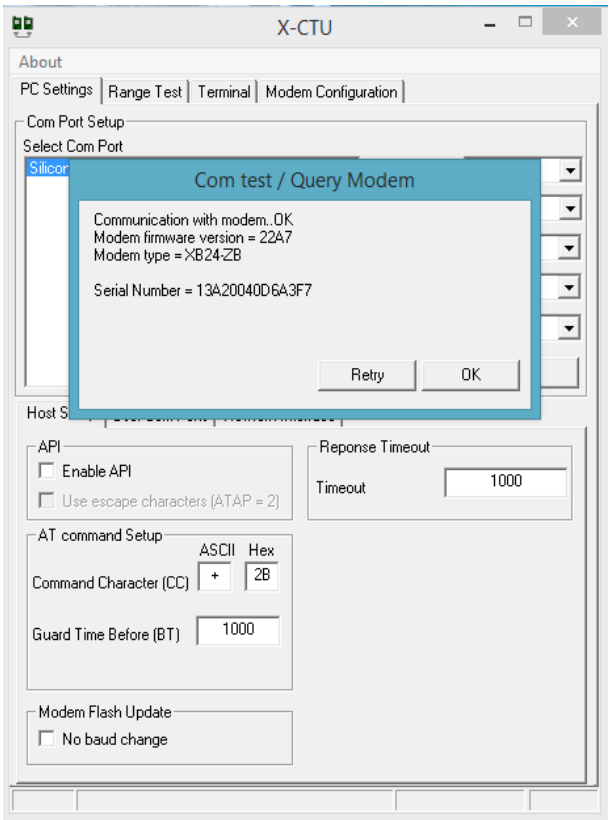

Figure 4.2: XBee Router PC settings

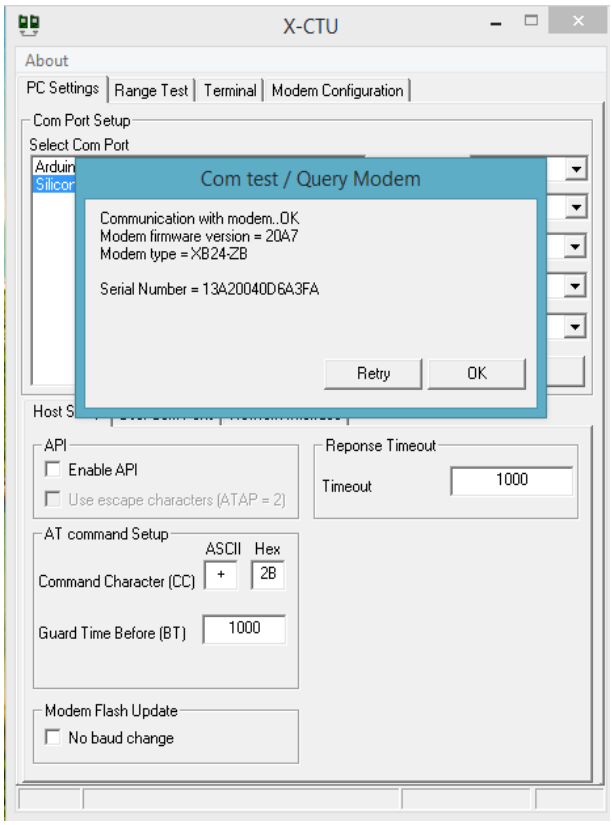

Figure 4.3: XBee Coordinator PC Settings

PC settings are very important and this is the place to test the true fate of any failing XBee. Set the baud rate and select the serial port to which XBee module has been connected. Baud rate of 9600bps has been set. No flow control and parity has been used. 1 stop bit and 8 data bits are selected. After testing query, we get a serial number corresponding to each XBee. The serial number of router XBee is the destination address of coordinator XBee and vice versa.

4.2.2 Modem Configuration

Firstly, XBee modules are read for their present configuration settings and then these settings are modified. One XBee module XB24-B is then configured as ZigBee Coordinator AT and another is configured as ZigBee Router AT. The AT mode has been selected because point to point communication is taking place. Hence, there is no requirement of API mode. Any random number eg.2411 is chosen as PAN ID. Both XBee modules are given same PAN ID because they have to communicate to each other. The destination address of each XBee is written.

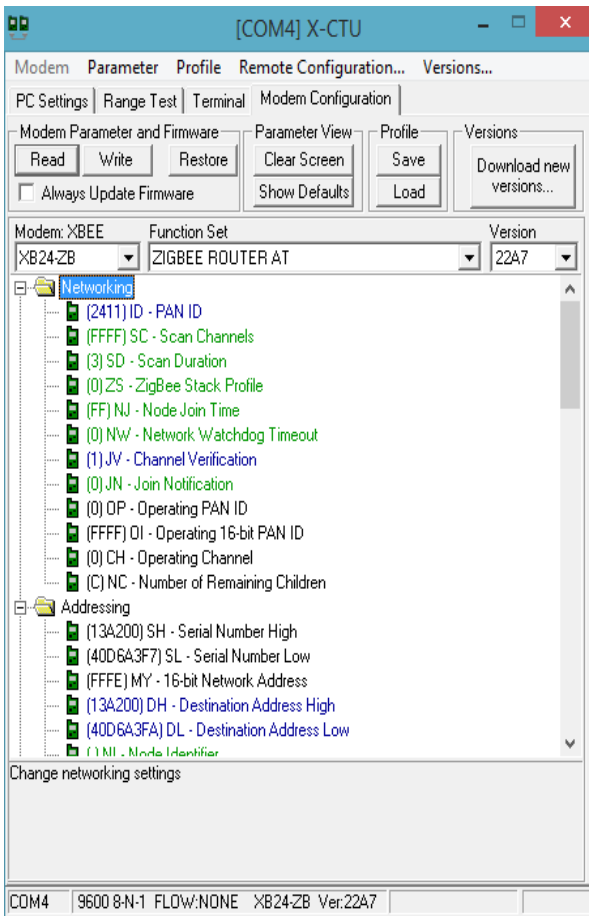

Figure 4.4: Router Config. Settings

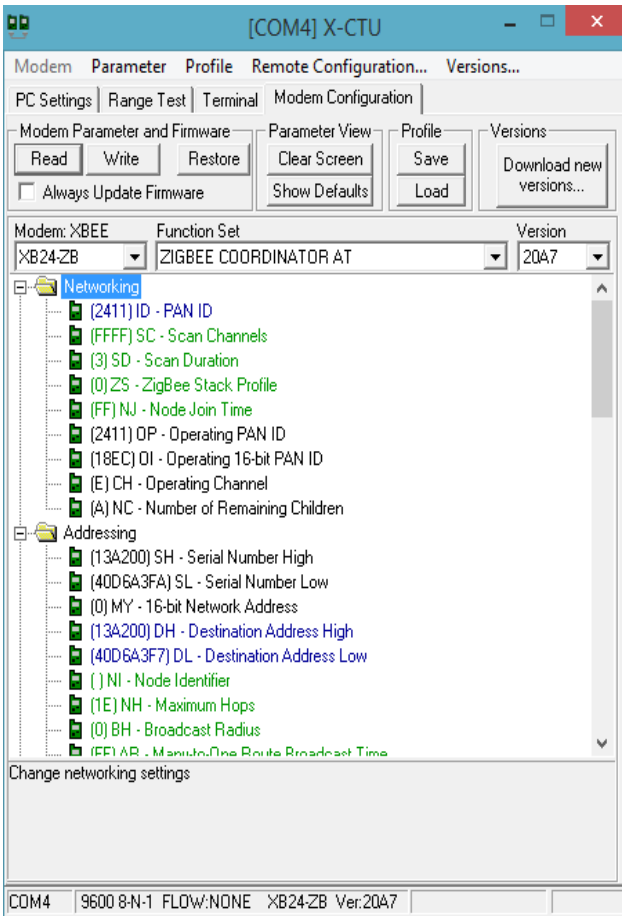

Figure 4.5: Coordinator Config. Settings

4.3 Module 3: Sensors Testing

Temperature sensor and PIR sensor are tested for their appropriate performance. Their testing circuits are implemented on the breadboard and readings are noted.

4.3.1 PIR Sensor Testing

Following circuit is implemented on the breadboard. 5V power supply, one led, one 220ohms resistance is used in this circuit. Proper connections are done and following readings are obtained. In this circuit, LED glows whenever any movement of any human being is detected by the PIR sensor. Readings are taken corresponding to human standing at different distances. Output voltage and on time of LED are noted.

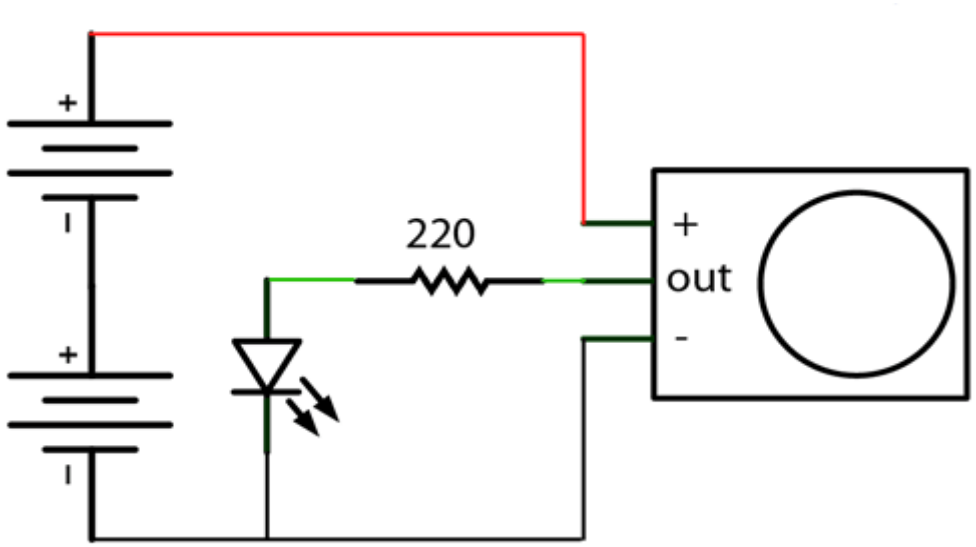

Figure 4.6: PIR Sensor Testing Circuit

Readings Obtained:

S. No.	Distance(cm)	Output Voltage(v)	On Time of LED (s)
1	8.5	3.13	2.0
2	15	3.12.	2.0
3	30	3.13	2.0
4	45	3.13	2.0

Table 4.1: PIR Sensor Testing Circuit Readings

4.3.2 Temperature Sensor Testing

For testing the temperature sensor, an experiment can be performed. In this experiment, V_c = 4 to 30V. Among these voltages, 5V or 12V are the typical values used. Ra should be $Vc/10^{-6}$. Hence, it can range from 80 KΩ to 600 KΩ. Voltmeter would be required to sense Vout. The output voltage is then converted to temperature by a simple conversion factor. The sensor has a sensitivity of 10mV/°C. Hence, a conversion factor which is the reciprocal, that is 100°C/V can be used.

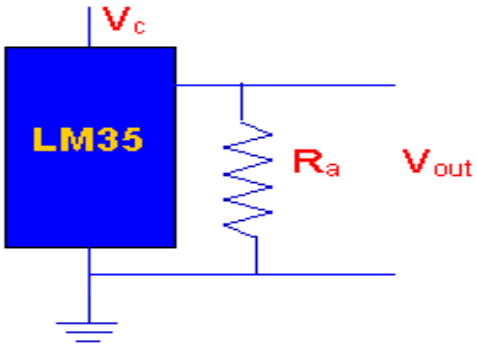

Figure 4.7: LM35 Testing Circuit

We have used V_c = 5V and R_a = 100KΩ. A high temperature object such as lighter is placed at different distances from the sensor and the output voltage is measured. The temperature corresponding to each voltage is then calculated.

Readings obtained:

Distance of Object (cm)	Output Voltage (V)	Temperature (°C)
2cm	0.2V	20^0
6cm	0.128V	12.8^0
10cm	0.1V	10^0
15cm	0.09V	9^0

Table 4.2: Temperature Sensor Testing Circuit Readings

4.4 Module 4: PC Controlled Wireless Robot

Robot is made wireless pc controlled by using XBee. The XBee coordinator is connected to pc via XBee USB adaptor and XBee router is connected to Arduino via XBee shield. Motor driver circuit is implemented on the breadboard and motors are connected to the Arduino by using this motor driver circuit. The left motor is connected to pwm pin12 and right motor is connected to pwm pin9 of the Arduino board. The code for wirelessly controlling the robotic vehicle with the help of 'w', 'a', 's' and 'd' keys of keyboard is written and compiled. Then it is uploaded on Arduino via USB cable. 'w' is used to move the robot in the forward direction, 'a' for moving it in left, 'd' for

right and ‘s’ for backward motion. Different serial commands for serial communication with XBee and digital input/output commands for motor interfacing are used in the Arduino code.
The code segment for this module is as follows:

```
int myData = 0;
const int lmotorpin = 12;
const int rmotorpin = 9;
void setup()
{
   Serial.begin(9600); // Baud rate of 9600 is set
   pinMode(lmotorpin,OUTPUT);  // Motor Pins are set as ouput pins
   pinMode(rmotorpin,OUTPUT);
}
void loop()
{
   if(Serial.available() > 0)     // If data is available on the serial port
   {
         myData = Serial.read();  // Read the data
         if(myData == 'w')  // For forward movement
         {
              digitalWrite(lmotorpin,HIGH);   // Write data on the motor pins
              digitalWrite(rmotorpin,HIGH);
              Serial.write(myData);
         }
         if(myData == 'a')  // For turning left
         {
              digitalWrite(lmotorpin,LOW);
              digitalWrite(rmotorpin,HIGH);
              Serial.write(myData);
         }
         if(myData == 's') // For moving in backward direction
         {
              digitalWrite(lmotorpin,LOW);
              digitalWrite(rmotorpin,LOW);
              Serial.write(myData);
```

```
        }
        if(myData == 'd')  // For turning right
        {
            digitalWrite(lmotorpin,HIGH);
            digitalWrite(rmotorpin,LOW);
            Serial.write(myData);
        }
        else
        {
            Serial.write(myData);
        }
    }
    delay(1000);
}
```

4.5 Module 5: Assembling all the sensors and Final Implementation

All the sensors are interfaced with Arduino board by writing the corresponding code. The distance measurements, temperature measurements and PIR sensor output signals are sent to XBee router connected to the Arduino which are then sent wirelessly to XBee coordinator connected to the pc. Hence, all the readings are obtained on pc wirelessly. The sensors are interfaced as follows:

- **Ultrasonic Sensor**

Trigger pin of sensor is connected to pin 2 and echo pin is connected to pin 4 of the Arduino board. Firstly, a low pulse of 2 microseconds is sent to the trigger pin and then trigger pin is made high for 10 microseconds. Then, the time duration until echo pin is high is recorded and converted to distance by using mathematical calculations. This calculated distance is then sent to serial port for wireless transmission to pc.

- **Temperature Sensor**

The output pin of the temperature sensor is connected to pin 0 of Arduino board. The data on this pin is read and converted to temperature which is then sent to the serial port for transmission to pc.

- **PIR Sensor**

The output pin of PIR sensor is connected to pin 6 of Arduino board. Firstly PIR sensor is calibrated by using a loop and a delay function. Then PIR output pin is monitored for motion detection.

The arduino code is as follows:

```
float temp;
const int tempPin = 0;

int myData = 0;
const int lmotorpin = 12;
const int rmotorpin = 9;

int pirData = 0;
boolean takeLowTime;
const int pirPin = 6;
const int ledPin = 13;
boolean lockLow = true;
long unsigned int lowIn;
int calibrationTime = 20;
long unsigned int pause = 5000;
const int trigPin = 2;
const int echoPin = 4;

void setup()
{
        // put your setup code here, to run once:
        Serial.begin(9600);
        pinMode(lmotorpin,OUTPUT);
        pinMode(rmotorpin,OUTPUT);
        pinMode(trigPin, OUTPUT);
        pinMode(echoPin, INPUT);
        pinMode(pirPin, INPUT);
        pinMode(ledPin, OUTPUT);
        //Calibration Of PIR Sensor
        digitalWrite(pirPin, LOW);
        Serial.print("Calibrating Sensor ");
        for(int i = 0; i < calibrationTime; i++)
        {
                Serial.print(".");
                delay(1000);
```

```
        }
        Serial.println("Sensor Active");
        delay(50);
}

void loop()
{
        long duration, inches, cm;
        //Sending Pulse To The Trigger Pin
        digitalWrite(trigPin, LOW);
        delayMicroseconds(2);
        digitalWrite(trigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(trigPin, LOW);

        //Read The Data On The Echo Pin
        duration = pulseIn(echoPin, HIGH);

        //Convert The Time Into Distance
        inches = microsecondsToInches(duration);
        cm = microsecondsToCentimeters(duration);

        //Sending Data Back To The Terminal
        Serial.println();
        Serial.print("DISTANCE OF OBJECT = ");
        Serial.print(inches);
        Serial.print("inches, ");
        Serial.print(cm);
        Serial.print("cm");
        Serial.println();

        //Temperature Sensor Interface
        temp = analogRead(tempPin);
        temp = temp * 0.48828125;
        Serial.print("TEMPRATURE = ");
        Serial.print(temp);
        Serial.print("*C");
```

```
 Serial.println();

 //PIR Sensor Interface
 if(digitalRead(pirPin) == HIGH)
{
        digitalWrite(ledPin, HIGH);
        if(lockLow)
        {
                lockLow = false;
                Serial.println("---");
                Serial.print("MOTION DETECTED AT: ");
                Serial.print(millis()/1000);
                Serial.println(" sec");
                delay(50);
        }
        takeLowTime = true;
}
if(digitalRead(pirPin) == LOW)
{
        digitalWrite(ledPin, LOW);
        if(takeLowTime)
        {
                lowIn = millis();
                takeLowTime = false;
        }
        if(!lockLow && millis() - lowIn > pause)
        {
                lockLow = true;
                Serial.print("MOTION ENDED AT: ");      //output
                Serial.print((millis() - pause)/1000);
                Serial.println(" sec");
                delay(50);
        }
}

if(Serial.available() > 0)
```

```
        {
            myData = Serial.read();
            if(myData == 'w')
            {
                digitalWrite(lmotorpin,HIGH);
                digitalWrite(rmotorpin,HIGH);
                Serial.write(myData);
            }
            if(myData == 'a')
            {
                digitalWrite(lmotorpin,LOW);
                digitalWrite(rmotorpin,HIGH);
                Serial.write(myData);
            }
            if(myData == 's')
            {
                digitalWrite(lmotorpin,LOW);
                digitalWrite(rmotorpin,LOW);
                Serial.write(myData);
            }
            if(myData == 'd')
            {
                digitalWrite(lmotorpin,HIGH);
                digitalWrite(rmotorpin,LOW);
                Serial.write(myData);
            }
            else
            {
                Serial.write(myData);
            }
        }
        delay(1000);
}
```

```
long microsecondsToInches(long microseconds)
{
        // There are 73.746 microseconds per inch (i.e. Sound travels at 1130 feet/sec
        // This gives the distance travelled by the wave, to and fro so we divide by 2 for result
        return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds)
{
        // The speed of sound is 340 m/s or 29 microseconds per centimeter.
        return microseconds / 29 / 2;
}
```

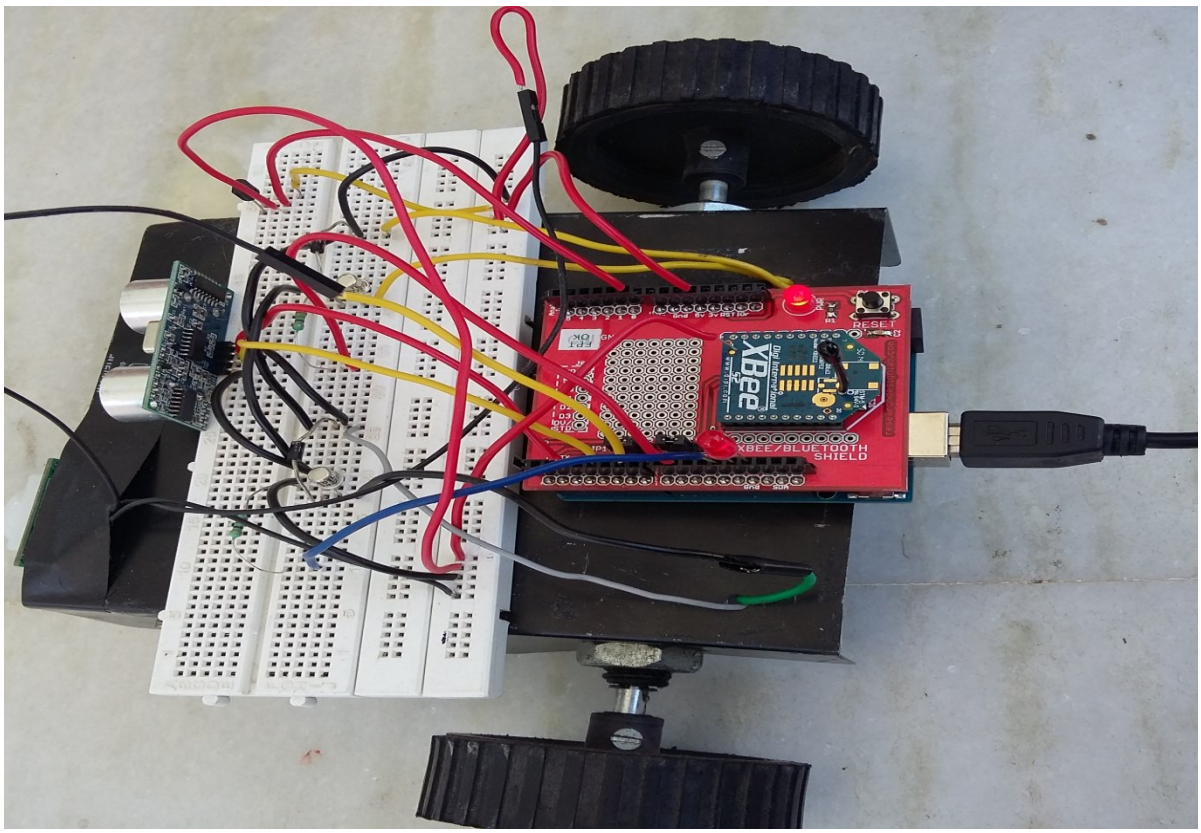

Figure 4.8: Top view of Robot

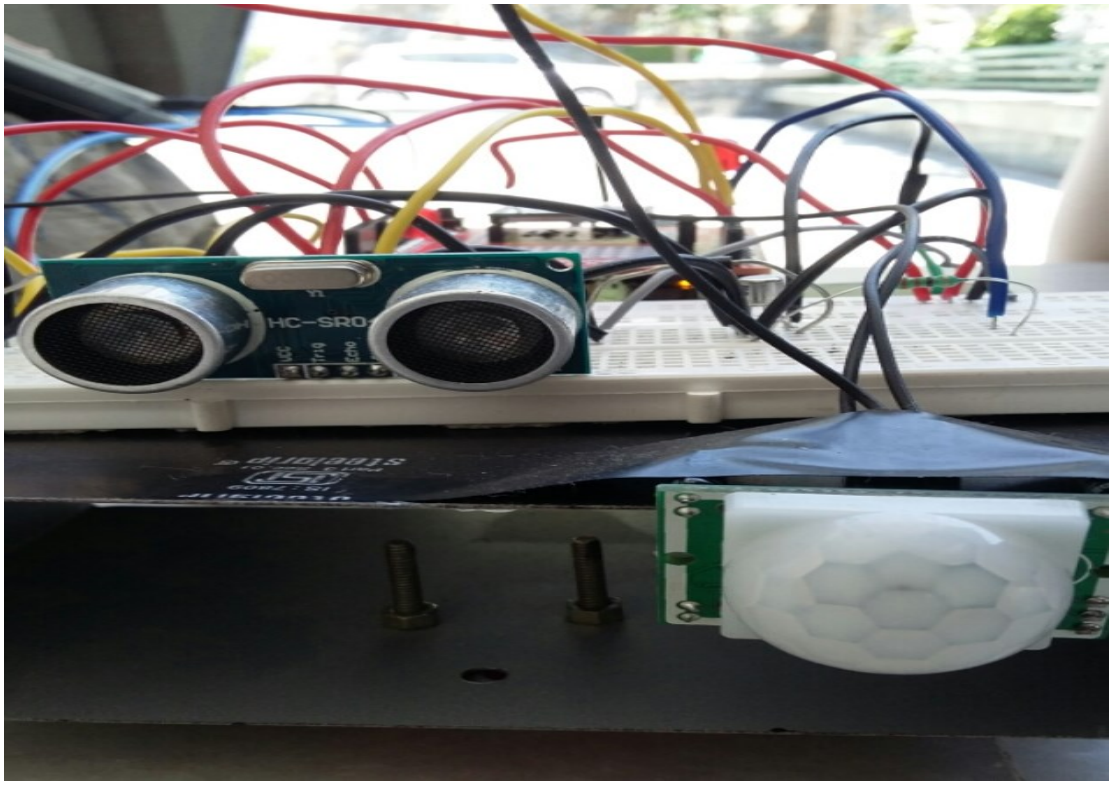

Figure 4.9: Front view of Robot

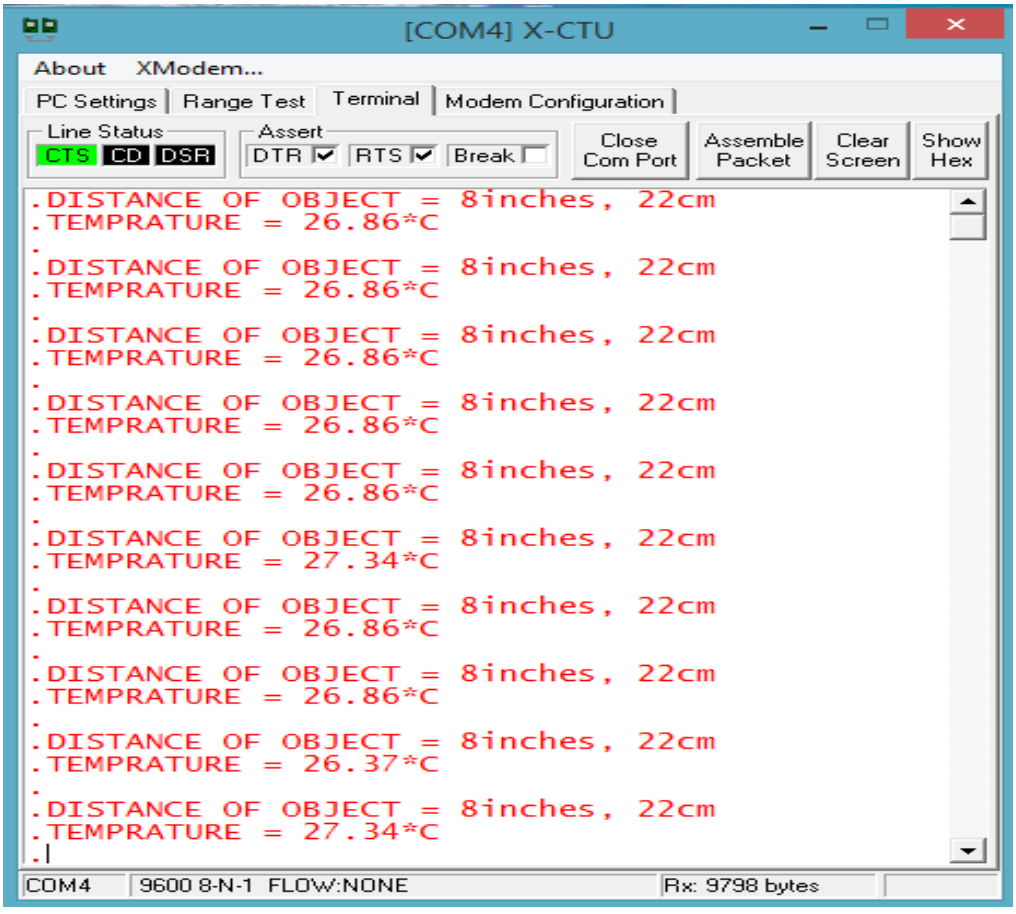

Figure 4.10: Terminal at Personal Computer

4.6 Problems Faced and Challenges

We encountered many problems and challenges while implementing the robotic system practically. Major problems constituted the calibration of the sensors used in this project. Sensors were firstly tested for their detection range and sensitivity by taking readings at different distances and angles. Then, only it was possible to use them effectively in the project. For example: PIR sensor required the calibration time of at least 10s – 60s. In our project, we have used 20s as its calibration time.

In addition to this, we also faced problem while interfacing motor with Arduino. The motor interfacing circuit was then used to prevent the Arduino from getting damaged. Firstly, we had used AVR development board named xboard v2.0 for the robot. The XBee interfacing was posing a challenge for us because of its lesser compatibility with XBee modules. Thus, we later decided to change the board and hence we moved to using Arduino board in our project because of its flexibility with all the components we were using and interfacing of Arduino with sensors and XBee was also quite easy.

Since XBee could not be interfaced with Arduino directly because of its lesser pin spacing, we used Arduino XBee adaptor board to resolve this problem. After facing all these challenges and finding the best possible solution for every problem, we were able to design and implement the project practically and effectively. Though it was time consuming to resolve all these problems, but this resolving phase provided us with a better understanding of various concepts involved in our project.

CHAPTER 5

Conclusion and Future Scope

5.1 Conclusion

On implementing the robotic vehicle for detecting human presence, we can assert with confidence that such systems can be used in variety of different applications. This wireless pc controlled robot can be efficiently used in disaster prone areas. Hence, many lives can be saved in a short duration which becomes time consuming and unaffected if done manually. In addition to this application, there can be many other aspects in which the robot can turn out to be efficient and useful.

Nowadays, human has become too busy and is unable to find time even to switch off the lights wherever not necessary. Our robot can be used to implement an intelligent lightning system in which lights get switched on automatically on detecting human presence and get turned off when there are no humans present. Brightness of the lights can also be controlled depending upon the requirements. Hence, electrical energy can be saved very effectively.

Another area of application is security system. Due to increasing number of crime and burglary, the need of security system is very essential. The security system that monitors the area

throughout the time and reacts effectively is required. This robotic vehicle can be used to detect the intruders and terrorists in the case of terrorist attacks. This system also provides us with amazing features of distance measurement and temperature measurement. Because of being PC controlled and wireless system, the robot is very portable and efficient. There is always a need to modify and add features according to the growing technology to compete the market. Even though, there are certain limitations to this system, but they can be overcome by making certain changes and enhancements to this robotic vehicle.

5.2 Future Scope

There are certain enhancements which would increase the efficiency of this robotic vehicle. Camera module can be used to record and analyse different conditions. This vehicle can be improved by using high range sensors and high capacity motors. Some more sensors like mobile phone detector, metal detector etc. can be implemented to make this vehicle more effective. More number of PIR sensors, ultrasonic sensors and temperature sensors can be mounted on the robot to increase the field of detection.

The robot can be designed such that it can detect stationary humans also. For this improvement, high quality sensors will be required. The system can be made more autonomous by implementing different path algorithms which will guide the robot in different conditions. This will help in reducing the manual work on pc. Robot can be modified such that it can become a fire fighting robot which will be able to enter into high temperature areas and thus will increase the human detection capability. These additions would make the system more complete and more capable of being able to work in any kind of environment.

References

[1] S.P Vijayaragavan, Hardeep Pal Sharma, Guna sekar.C.H, S.Adithya Kumar, June 2013, "Live Human Detecting Robot for Earthquake Rescue Operation", Vol 02

[2] Trupti B. Bhondve, Prof. R.Satyanaraya, Prof. Moresh Mukhedkar, June 2014, "Mobile Rescue Robot for Human Body Detection in Rescue Operation of Disaster", Vol 3

[3] Martin Hebel, George Bricker with Daniel Harris, 2010, "Getting Started with XBee RF Modules"

[4] http://www.sensorcentral.com/photoelectric/ultrasonic01.php

[5] http://arduinobasics.blogspot.in/2012/11/arduinobasics-hc-sr04-ultrasonic-sensor.html

[6] http://www.instructables.com/id/Arduino-Basics-PIR-Sensor/

[7] http://www.instructables.com/id/Changing-Xbee-Baud-Rates/

[8] https://learn.adafruit.com/downloads/pdf/pir-passive-infrared-proximity-motion-sensor.pdf

[9] http://www.instructables.com/id/Arduino-Temperature-Sensor-Interfacing-LM35-THE-EA/

[10] https://learn.adafruit.com/adafruit-arduino-lesson-13-dc-motors/overview

[11] https://www.sparkfun.com/products/10414

[12] http://www.digi.com/pdf/ds_xbeezbmodules.pdf

[13] http://www.arduino.cc/en/Main/ArduinoBoardUno

[14] http://www.instructables.com/id/Intro-to-Arduino/step2/Arduino-Uno-Features/

[15] http://knowledge.digi.com/articles/Knowledge_Base_Article/X-CTU-XCTU-software