A FUZZY LOGIC MICRO-CONTROLLER ENABLED SYSTEM FOR THE MONITORING OF MICRO CLIMATIC PARAMETERS OF A GREENHOUSE

A thesis submitted in partial fulfilment of Mtech degree in electrical engineering

at the University of South Africa.

Author: Malusi Sibiya

Student number: 43447619

Supervisor: Dr Sumbwanyambe

Tittle of dissertation:

A FUZZY LOGIC MICRO-CONTROLLER ENABLED SYSTEM FOR THE MONITORING OF MICRO   CLIMATIC PARAMETERS OF A GREENHOUSE.

Key terms:

Fuzzy logic inference; Membership functions; Green house; Wireless networks; XBee; ZigBee; MATLAB; Raspberry pi; Python; XCTU

DECLARATION

Name: MALUSI SIBIYA

Student number: 43447619

Degree: MTech   ELECTRICAL

Exact wording of the title of the dissertation or thesis as appearing on the copies submitted for examination:

A FUZZY LOGIC MICRO-CONTROLLER ENABLED SYSTEM FOR THE

MONITORING OF MICRO   CLIMATIC PARAMETERS OF A GREENHOUSE

I declare that the above dissertation/thesis is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references.

[illegible]

_________ ______	9 October_2017
SIGNATURE	DATE

ABSTRACT

Motivation behind this master dissertation is to introduce a novel study called " A fuzzy logic micro-controller enabled system for the monitoring of micro-climatic parameters of a greenhouse" which is capable of intelligently monitoring and controlling the greenhouse climate conditions in a preprogrammed manner.

The proposed system consists of three stations: Sensor Station, Coordinator Station, and Central Station. To allow for better monitoring of the climate condition in the greenhouse, fuzzy logic controller is embedded in the system as the system becomes more intelligent with fuzzy decision making. The sensor station is equipped with several sensor elements such as MQ-7 (Carbon monoxide sensor), DHT11 (Temperature and humidity sensor), LDR (light sensor), grove moisture sensor (soil moisture sensor). The communication between the sensor station and the coordinator station is achieved through XBee wireless modules connected to the Arduino Mega and the communication between coordinator station and the central station is also achieved via XBee wireless modules connected to the Arduino Mega.

The experiments and tests of the system were carried out at one of IKHALA TVET COLLEGE’s greenhouses that is used for learning purposes by students studying agriculture at the college. The purpose of conducting the experiments at the college’s green house was to determine the functionality and reliability of the designed wireless sensor network using ZigBee wireless technology. The experiment result indicated that XBee modules could be used as one solution to lower the installation cost, increase flexibility and reliability and create a greenhouse management system that is only based on wireless nodes. The experiment result also showed that the system became more intelligent if fuzzy logic was used by the system for decision making.

The overall system design showed advantages in cost, size, power, flexibility and intelligence. It is trusted that the results of the project will give the chance for further research and development of a low cost greenhouse monitoring system for commercial use.

ACKNOWLEDGEMENTS

With God in my prayers, my mind was relaxed and open in all the ups and downs of this study. I would first like to thank my thesis advisor Dr. M. Sumbwanyambe of the school of Electrical engineering and Mining at University of South Africa. The door to Dr. M. Sumbwanyambe's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work, but steered me in the right direction whenever he thought I needed it. I would also like to acknowledge Patrick Mahlangu who was also a student supervised by Dr. M. Sumbwanyambe as the second reader of this dissertation, and I am gratefully indebted to his very valuable comments on this dissertation.

Finally, I must express my very profound gratitude to my parents, IKHALA TVET COLLEGE and to my wife for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation. This accomplishment would not have been possible without them. Thank you.

Author

Malusi Sibiya

TABLE OF CONTENTS

LIST OF FIGURES 10

LIST OF TABLES 14

GLOSSARY 15

CHAPTER 1: Introduction 16

Background to study 16

Fuzzy logic and smart agriculture 16

1.3 The research problem 19

1.4 The objectives of the study 19

1.5 Methodology 21

1.6 Research design 22

1.7 Data analysis and interpretation 22

1.8 Data sources 23

1.9 Data collection techniques 23

1.10 Reliability issues and validity 24

1.11. Dissertation outline 24

CHAPTER 2: Wireless technologies for smart agriculture 25

2.1. Introduction 25

2.2 Wireless sensor networks in greenhouse monitoring 25

2.3. Common Hardware Platform for most wireless sensor networks 26

2.4 Raspberry Pi platform 29

2.5Arduino microcontroller platform 31

2.6 Types of wireless sensor monitoring gadgets 31

2.6.1 Win land Enviro Alert... 31

2.6.2 Watchdog Wireless Crop Monitor 33

2.7 Routing protocols for wireless sensor networks 34

2.8 ZigBee Protocol Architecture 35

2.9 Power management in WSNs 36

2.10 Related work 37

2.11 Chapter summary 39

CHAPTER 3: Methodology 40

3.1. Introduction 40

3.2 Block diagrams for configuring XBee radios 41

3.3 Sensor station design 45

3.3.1 Putting it altogether for the sensor station 47

3.3.2 Sensor data analysis and complete circuit diagram of the sensor station 48

Specifications of Grove moisture sensor 49

3.3 Coordinator station design 56

3.3.1. LCD and Bluetooth module functions in the coordinator station 57

3.3.2. Function of the Led in the coordinator station 62

3.3.3. Function of the relay module in the coordinator station 62

3.4 Central station design 64

3.5 Arduino libraries 65

3.6 Python libraries 66

3.7 Experiment 1: Measuring soil moisture content using grove moisture sensor 67

3.8 Experiment 2: Measuring temperature and humidity using DHT11 sensor 68

3.9 Experiment 3: Measuring Carbon monoxide content using MQ7 gas sensor 69

3.11 Experiment 5: Demonstrate fuzzy logic control of temperature, humidity, moisture content, Carbon monoxide concentration and light intensity 70

3.12 Chapter summary 70

CHAPTER 4: Presentation of experiments results and their analysis 71

4.1 Introduction 71

4.2 Experiment 1 results: Analysis of moisture sensor data plots 71

4.3 Experiment 2 results: Analysis of humidity and temperature data plots 74

Analysis of temperature sensor data plots 74

Analysis of humidity sensor data plots 75

4.4 Experiment 3 results: Analysis of carbon monoxide data plots 76

4.5 Experiment 4 results: Analysis of light sensor(LDR) data plots 78

4.6 Experiment 5: Fuzzy logic experiment results 80

4.6.1 Fuzzy logic results of temperature and humidity 80

4.6.2 Fuzzy logic results of carbon monoxide (CO) gas 95

4.6.3 Fuzzy logic results of soil moisture content 98

4.6.4 Fuzzy logic results of light intensity 102

4.7 Chapter summary 104

CHAPTER 5: Summary of the study, Conclusions, Implications and Recommendations 105

5.1 Introduction 105

5.2 Summary of the study 105

5.3 CONCLUSIONS AS PER RESEARCH OBJECTIVES 108

5.3.1 General objective 108

5.3.2 Research objective1 108

5.3.3 Research objective2 108

5.3.4 Research objective3 108

5.3.5 Research objective 4 109

5.4 Recommendations 109

5.5 Limitations of this study 110

5.6 Future research 110

REFERENCES 111

APPENDIX A 117

APPENDIX B 120

APPENDIX C 138

APPENDIX D 142

LIST OF FIGURES

Figure 1.1: Fuzzy logic controller 17

Figure 1.2: Block diagram of the entire system 21

Figure 1.3: Research design to be followed when designing a system 22

Figure 2.1: ZigBee Pair, Star, Mesh, and Cluster tree topologies 26

Figure 2.2: Functional piece outline of ViFDAQ demonstrating the microcontroller and firmware center, analog and digital IOs, installed information and program memory, locally available sensors, and interfaces to outer sensors, host or sensor networks [12] 27

Figure 2.3: Flowchart for programming a measuring task for the ViFDAQ sensor platform [12] 28

Figure 2.4: XBee radios: Regular and Pro types 29

Figure 2.5: Raspberry Pi configured for XBee wireless radio 30

Figure 2.6: Arduino adapter hack for XBee wireless radio 31

Figure 2.7: Win land Enviro Alert 32

Figure 2.8: Watchdog Wireless Crop Monitor 34

Figure 2. 9: ZigBee protocol architecture 35

Figure 3.1: Block diagram of the entire system 41

Figure 3.2: XBee router configuration 42

Figure 3.3: XBee coordinator configuration 42

Figure 3.4: Configuration of XBee router and coordinator on XCTU platform 43

Figure 3.5: Communication test between XBee radio modules configured as router and coordinator 44

Figure 3.6: Wireless Proto shield 45

Figure 3.7: Arduino Mega 2560 46

Figure 3.8: Series2 XBee radio with its connection layout 46

Figure 3.9: The Arduino with the XBee and Wireless Proto Shield installed 47

Figure 3.10: Arduino IDE 48

Figure 3.11: Grove moisture sensor 49

Figure 3.12: DHT11 Sensor 50

Figure 3.13: LDR sensor 51

Figure 3.14: Characteristics curve of the LDR 51

Figure 3.15: MQ7 gas sensor 52

Figure 3.16: Sensitivity characteristics curve of the MQ-7 54

Figure 3.17: Complete circuit diagram of the sensor station 55

Figure 3.18: Designed sensor station 56

Figure 3. 19: The Arduino with XBee and wireless proto-shield installed 57

Figure 3.20: LCD display at power reset 58

Figure 3.21: LCD displays information about data collected by sensor 58

Figure 3.22: HC-05 Bluetooth terminal 59

Figure 3.23: LCD display if there is no CO gas detected 60

Figure 3.24: LCD display if CO gas is detected 61

Figure 3.25: HC-05 Typical application circuit 61

Figure 3.26: Circuit diagram of the coordinator station 63

Figure 3.27: Designed coordinator station 63

Figure 3.28: Circuit diagram of the central station 64

Figure 3.29: Designed central station 65

Figure 3.30: Grove moisture sensor inserted in soil to measure soil moisture content 67

Figure 3.31: DHT11 sensor measuring temperature and humidity in a greenhouse 68

Figure 3.32: MQ7 sensor used to measure carbon monoxide 69

Figure 3.33: Light sensor(LDR) used to measure light intensity 69

Figure 4. 1: Moisture sensor plots 73

Figure 4. 2: Temperature and Humidity plots 75

Figure 4. 3: Carbon monoxide data plots 77

Figure 4. 4: Light intensity data plots... 79

Figure 4. 5: Temperature input variable with cold, normal and hot membership functions 80

Figure 4. 6: Humidity input variable with low, optimal and high membership functions 82

Figure 4. 7: Output variable called “reference output” with phase1-phase9 membership functions 82

Figure 4. 8: Surface view Z-output of temperature input y-axis and humidity input x-axis...... 84

Figure 4. 9: Surface view Z-output of humidity input y-axis and temperature input x-axis...... 84

Figure 4. 10: Output results when rule 1 was invoked (temperature and humidity fuzzy logic analysis)... 85

Figure 4. 11: Output results when rule 2 was invoked (temperature and humidity fuzzy logic analysis)... 86

Figure 4. 12: Output results when rule 3 was invoked (temperature and humidity fuzzy logic analysis)... 87

Figure 4. 13: Output results when rule 4 was invoked (temperature and humidity fuzzy logic analysis)... 88

Figure 4. 14: Output results when rule 5 was invoked (temperature and humidity fuzzy logic analysis)... 89

Figure 4. 15: Output results when rule 6 was invoked (temperature and humidity fuzzy logic analysis) 90

Figure 4. 16: Output results when rule 7 was invoke (temperature and humidity fuzzy logic analysis) 91

Figure 4. 17: Output results when rule 8 was invoked (temperature and humidity fuzzy logic analysis) 92

Figure 4. 18: Output results when rule 9 was invoked (temperature and humidity fuzzy logic analysis) 93

Figure 4. 19: CO input variable with “safe level” and “beyond maximum” membership functions. 95

Figure 4. 20: Output variable called CO_2 injector with “CO_2 injector on” and “CO_2 injector off” membership functions. 96

Figure 4. 21: Surface view Z-output of carbon monoxide input x-axis 96

Figure 4. 22: Output results when rule 1 was invoked (CO gas fuzzy logic analysis) 97

Figure 4. 23: Output results when rule 2 was invoked (CO gas fuzzy logic analysis) 98

Figure 4. 24: Soil moisture input variable with dry, damp and wet membership functions 98

Figure 4. 25: Output variable called sprinkler with sprinkler on and sprinkler off membership functions. 99

Figure 4. 26: Surface view Z-output of soil moisture input x-axis 99

Figure 4. 27: Output result when rule 1 was invoked (soil moisture fuzzy logic analysis) 100

Figure 4. 28: Output results when rule 2 was invoked (soil moisture fuzzy logic analysis) 101

Figure 4. 29: Output result when rule 3 was invoked (soil moisture fuzzy logic analysis) 101

Figure 4. 30: Light intensity input variable with “not dark” and “dark” membership functions 102

Figure 4. 31: Output variable called bulb and its membership functions................................ 102

Figure 4. 32: Surface view Z-output of light intensity input x-axis.. 103

Figure 4. 33: Output results when rule 1 was invoked (light intensity fuzzy logic analysis) .. 103

Figure 4. 34: Output results when rule 2 was invoked (light intensity fuzzy logic analysis) .. 104

LIST OF TABLES

Table 1.1: Methods of displaying data collected from the sensors 23

Table 3.1: Router and Coordinator configurations in the XCTU program 44

Table 3.2: Specifications of DHT11.. 50

Table 3.3: Specifications of MQ-7 Carbon monoxide sensor 53

Table 3.4: Coordinator station components.. 56

Table 3.5: Led specifications.. 62

Table 4.1: Specifications of grove moisture sensor.. 72

Table 4.2: Moisture sensor analog readings and corresponding voltages as read from the moisture sensor data plots. 72

Table 4.3: Temperatures in degrees Celsius as read from temperature sensor data plots of...... 74

Table 4.4: Humidity percentages as read from humidity sensor data plots of Figure 4.2.......... 75

Table 4.5: Carbon monoxide analog readings and corresponding voltages as read from gas sensor data plots of Figure 4.3.. 76

Table 4.6: Light intensity analog readings and corresponding voltages as read from light sensor data plots of Figure 4.4 78

GLOSSARY

Arduino – Microcontroller board that utilizes AVR chip.

XBee – A wireless radio from a company called DIGIKEY.

XCTU - A software for programming XBee radios.

ZigBee – A protocol for operating XBee.

WSN – A wireless sensor network that is formed communication of XBee radios.

Raspberry pi – A microcomputer that uses Linux and consists of input/output ports (IO ports).

Fuzzy logic-A way of expressing the degree of membership between binary 0 and 1.

MATLAB- A program used to simulate fuzzy logic experiments.

IDE- Integrated development environment.

DIGIKEY- A company that manufactures XBee radios.

Bluetooth AT mode - AT command mode allows the user to interrogate the Bluetooth module and also to change some of the settings such as the name, whether or not it operates in slave mode or master mode and the baud rate.

Key- A key is also known as a legend and is used to identify different graph plots plotted on the same system of axis.

GSM systems- Systems based on Global Systems for Mobile communication. These systems utilize SMS messages to report to the mobile phone of the user.

Leachate- Refers to liquids that migrate from the waste carrying dissolved. Leachate is basically any liquid that, in the course of passing through matter, extracts suspended solids or any other component of material through which it has passed.

CHAPTER 1: Introduction

Background to study

Developments in the field of wireless sensor networks (WSNs) as well as miniaturization of the sensor boards in many scientific fields can benefit automated agriculture systems by exploiting WSNs as an effective tool for application research [1]. A number of applications integrate WSNs. These applications include environmental monitoring, energy management in the building and monitoring of certain ecosystems to mention a few. As for environmental monitoring, agriculture has so far become one of the fields where WSNs had been used such as in the monitoring of greenhouses. Greenhouses are structures covered with transparent glass or plastic films specially designed to grow plants inside. Greenhouses can modify crop micro climate according to plant needs and therefore contribute to increase in crop productivity and produce quality. The degree of environmental control provided by a greenhouse varies with its characteristics and the technological sophistication of the equipment.

Fuzzy logic and smart agriculture

Normally in logic there is a series of statements which is either true or false. For example, a temperature of 25 degrees Celsius is true if the exact value is read and measured. What if the temperature is 24.5 degrees Celsius or even higher than 25 degrees Celsius such as 25.8? This will bring uncertainties. However, by means of fuzzy logic such a problem could be solved. In a clear sense, fuzzy logic is an intelligent system of multivalued logic. These values deal with uncertainty in most engineering applications by attaching degrees of certainty to a question that is logical. Most agricultural applications are based on fuzzy logic for decision making. In this study, the system will be based on Fuzzy logic control. Fuzzy logic control is chosen for the following reasons:

i. It can be used in designing of systems where it is impossible to get their models for conventional control.
ii. It can be used to mimic the experienced human operator to model the control system.
iii. It is easily being embedded in small microcontrollers and processors.
iv. Compared with conventional control, fuzzy logic is more accurate.

Figure 1.1 shows the block diagram of fuzzy logic controller.

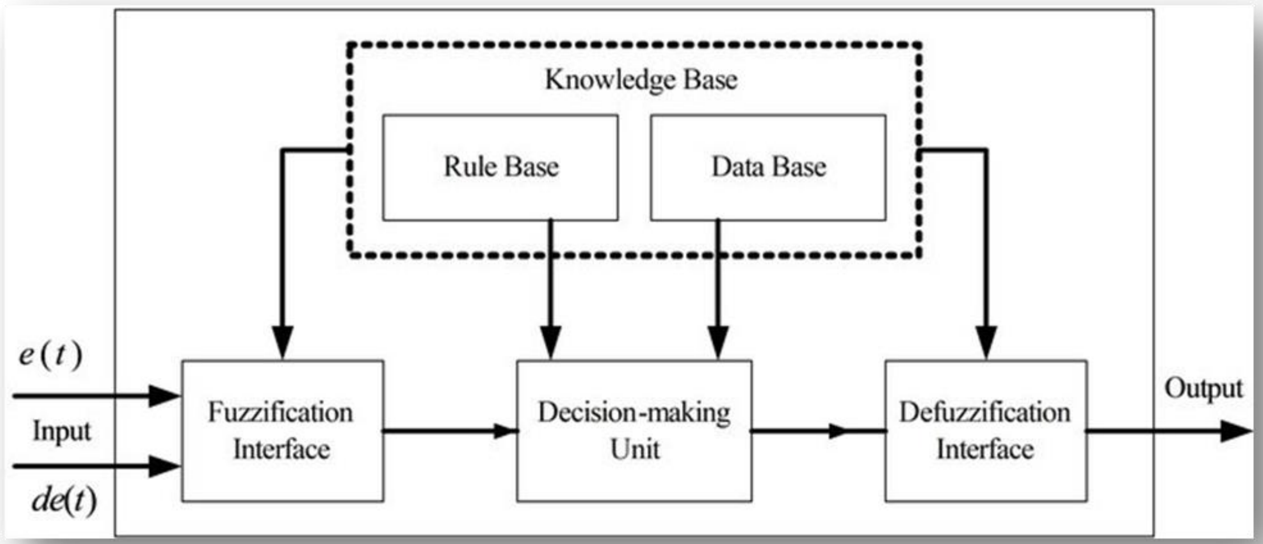


Figure 1.1: Fuzzy logic controller

Fuzzy logic has been used for a wide variety of applications. Among these applications, it has been used for irrigation control to achieve the intelligent decision making of a human operator. It determines when the irrigation should begin based upon the information from soil sensors. It can also learn the irrigation schedules by itself during on-line control process [2]. Literature points out that there are a lot of papers in which fuzzy logic has been applied. For example, Nandwana et al [3] used fuzzy logic control on a solar powered pump for irrigation of green vegetable plants. Others such as Wang et al [3] investigated the use of fuzzy logic in the lumber drying process. In the investigation, a conventional adaptive controller was implemented as well, and its performance was compared with that of fuzzy logic adaptive controller. The results were outstanding for fuzzy logic controller as it managed to overcome complex, non-linear, and time variant nature of drying process when conventional control was used. Human intervention in agricultural processes has so far been minimized by existing control systems. Generally, in such processes fuzzy logic has been implemented in the design of such intelligent systems and is renowned for its ability to lessen the complexity of certain systems without using mathematical models. As a matter of fact, Prema et al [4] actually designed a robot so as to overcome human intervention in the field of agriculture.

This agricultural robot with the composed controller would be controlled from any remote PC. The robot therefore reduced the human intervention needed and became advantageous and cost-effective. In agriculture, climatic conditions such as temperature, humidity, carbon dioxide, light and soil moisture matter a lot. Tentatively, it is worthwhile to point out at this stage that temperature is relative to humidity. Therefore, in designing fuzzy controlled systems one must design system that takes into account, if not all of the climatic conditions, temperature and humidity. For this reason, systems have been designed to control environmental temperatures and humidity. In most of these systems, humidity and temperature were used as input variables in the design of an intelligent dryer system for agriculture industry [5]. Another factor worth mentioning is light. Light from the sun is accountable for all life on the earth. Daylight fulfills the procedure of photosynthesis where plants change over carbon dioxide and water into sugars [6]. Plants utilize light in the scope of 400 to 700 nanometers. This range is normally alluded to as photo synthetically active radiation (PAR). Observing PAR is imperative to guarantee that plants are getting sufficient light for their photosynthesis. Fuzzy logic control is sometimes utilized to monitor PAR [7]. Another important factor in climatic conditions of plants is carbon dioxide (CO_2). Carbon dioxide is a standout among the most widely recognized results of living beings [8]. The gas itself is safe in low concentration, however in higher concentration it could be dangerous. Fuzzy logic control has been utilized to keep up carbon dioxide concentration in a greenhouse [7]. To monitor all of the above climatic conditions for better plant yields, lots of newer and rising technologies have been used. Among these rising wireless innovations, ZigBee was accounted for as a standout among the most used wireless advancements in cutting edge businesses [9]. ZigBee wireless innovation is fit for giving extensive scale on low power systems and gadgets that could keep running for quite a long time on modest batteries. Also, its ease and low system multifaceted nature qualities make it perfect for some agribusiness applications. For example, ZigBee has been used in greenhouse atmosphere control. In certain instances, fuzzy logic algorithm had been utilized as a part of the improvement of the system that comprises of slave hubs and a master station each of which is furnished with a ZigBee wireless miniaturized micro-controller [10].

1.3 The research problem

Environmental conditions that are appropriate happen to be necessary for optimum plant growth, improved crop yields, and efficient use of water and other resources. Automating data acquisition of the soil conditions and different climatic parameters that administer plant development enables data to be gathered at high recurrence with less work prerequisites. At present existing systems utilize LCD-based systems for keeping the user persistently informed of the conditions inside the green house. These systems are excessively expensive, cumbersome, hard to keep up and less acknowledged by unskilled labor. None, so far, in South Africa, to our best of knowledge has employed fuzzy logic and microcontrollers.

From research problem, the following research questions were developed:

i. What literature is available on the existing micro-climatic monitoring systems?
ii. What are the improved methods of wireless communication for the proposed system?
iii. How does fuzzy logic improve the proposed system compared to existing systems?
iv. What are the improved methods of data monitoring?
v. How will the proposed system be miniaturized?

1.4 The objectives of the study

The objective of this research is to design a system that monitors the climatic conditions of a greenhouse and automatically switch on and off the controlled outputs depending on the climatic conditions of a greenhouse. The system must be, easy to install, microcontroller-based circuit to monitor and record the values of temperature, humidity, soil moisture, carbon dioxide and sunlight of the natural greenhouse environment that are continuously modified and controlled in order to optimize them and achieve maximum plant growth and yield. The controller used will be a low power, cost efficient chip manufactured by ATMEL having 8KB of on-chip flash memory. It will communicate with the various sensor modules in real-time in order to control the light, humidity, temperature, soil moisture and carbon monoxide efficiently inside a greenhouse by actuating a fan, fogger, heater and lights respectively according to the necessary condition of the crops. An integrated Liquid crystal display (LCD) will also be used for real time display of data acquired from the various sensors. Also, the use of easily available components will reduce the manufacturing and maintenance costs. The design has to be quite flexible as the software can be changed any time. It can thus be tailor-made to the specific requirements of the user.

This makes the proposed system to be an economical, portable and a low maintenance solution for greenhouse applications, especially in rural areas and for small scale agriculturists. Since fuzzy logic controller is the proposed means of control, the system will be designed based on knowledge of an experienced operator and how climatic conditions must be monitored for the well-being of plants grown in a greenhouse.

The specific objectives of this study are:

i. To carry out extensive literature review on the existing systems.
ii. To determine, design and develop a low-cost microcontroller system for the monitoring of climatic parameters in a greenhouse using smart technology.
iii. The designed system must keep the user consistently informed of the conditions inside a greenhouse.
iv. To use fuzzy logic to optimize the control algorithms of a greenhouse monitoring system by use of wireless XBee radio modules and sensors.
v. To design a system with less installation of complex electric wires in order to save energy.
vi. To specify the necessary climatic parameters for humidity, temperature, soil moisture and sunlight for sensors.

All the research questions must be answered and ensure that all these feature are integrated in the intended system. MATLAB program will also be used in real time simulation of fuzzy logic experiments. ZigBee wireless protocol will be used to acquire data through sensors from different angles of a greenhouse hence forming a communication network. A complete block diagram of the research system is as shown in Figure 1.2.

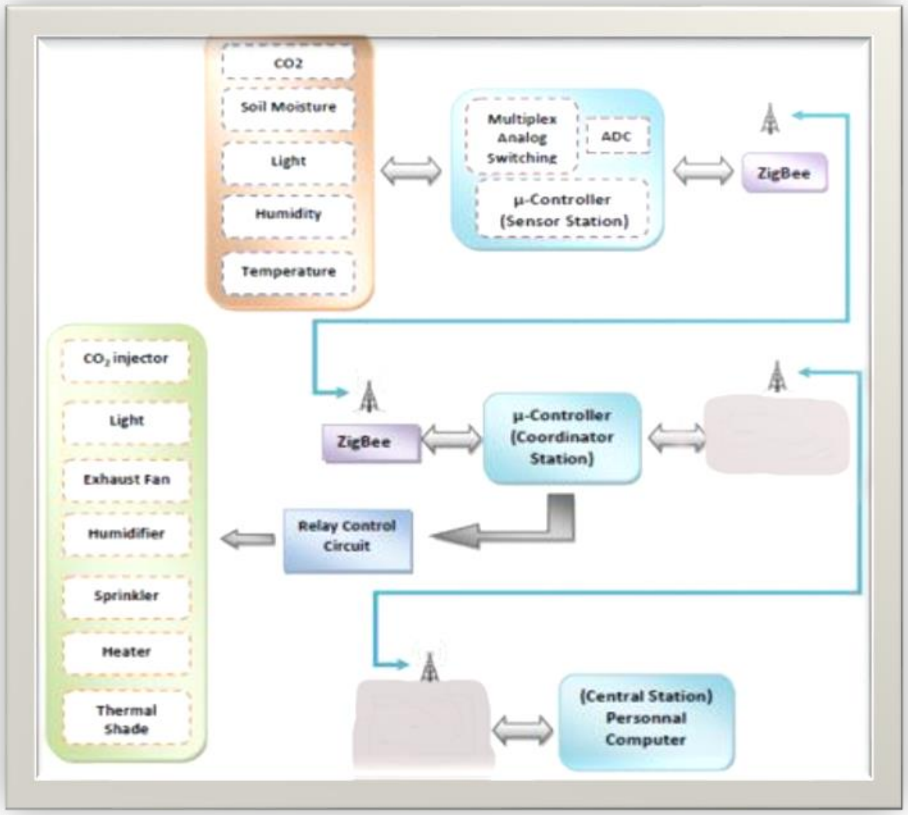


Figure 1.2: Block diagram of the entire system

1.5 Methodology

The research process followed in this study is a descriptive experimental and quantitative research. The control system will be designed and built to monitor the greenhouse without any human intervention. Different stages of wireless communication will be designed and tested for effective communication. This research system's results will be compared to results of a greenhouse monitored by humans for temperature control, humidity control, soil moisture and light intensity and carbon monoxide concentration. For this system, fuzzy logic inference for temperature and humidity is going to be achieved by means of fuzzy logic implemented in the Arduino microcontroller and the results simulated in MATLAB. There will be wireless communication of XBee radio modules to enable high probability data collection in the green house and complex wiring will be minimized.

1.6 Research design

The research design to be followed in this study is shown as in Figure 1.3. Design science research is paradigm that will be followed the design of the intended system.

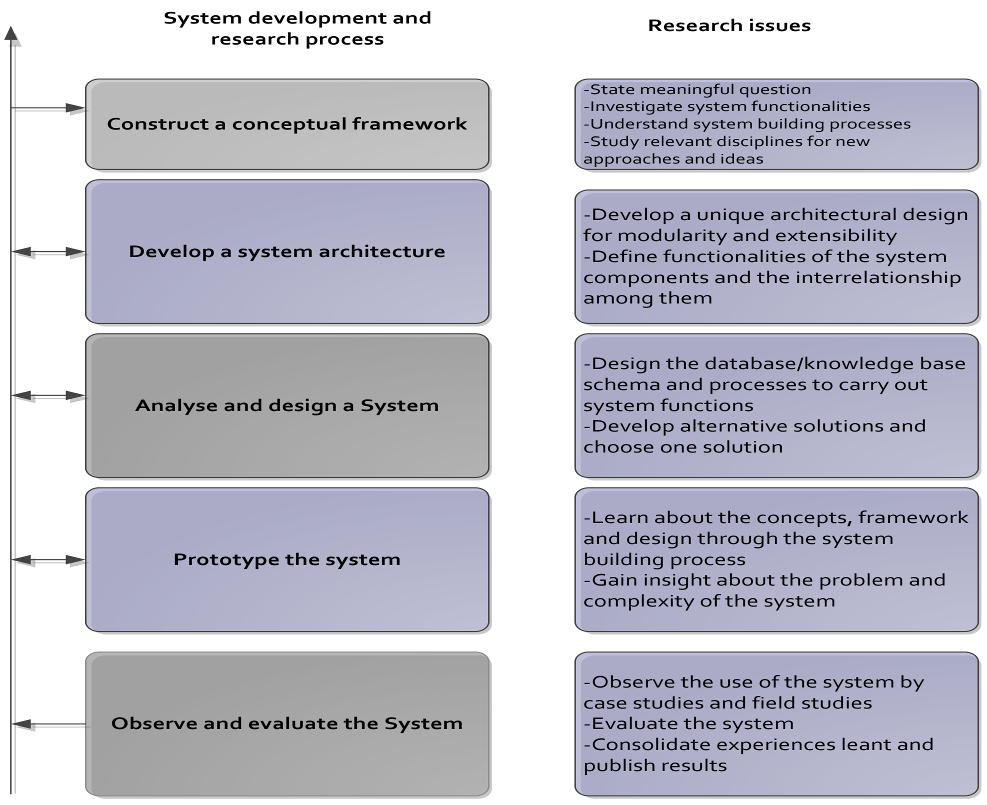

System development and research process
Research issues
Construct a conceptual framework
-State meaningful question
-Investigate system functionalities
-Understand system building processes
-Study relevant disciplines for new approaches and ideas
Develop a system architecture
-Develop a unique architectural design for modularity and extensibility
-Define functionalities of the system components and the interrelationship among them
Analyse and design a System
-Design the database/knowledge base schema and processes to carry out system functions
-Develop alternative solutions and choose one solution
Prototype the system
-Learn about the concepts, framework and design through the system building process
-Gain insight about the problem and complexity of the system
Observe and evaluate the System
-Observe the use of the system by case studies and field studies
-Evaluate the system
-Consolidate experiences leant and publish results

Figure 1.3: Research design to be followed when designing a system

1.7 Data analysis and interpretation

Data will be analyzed using different methods. The best method will be to link the Arduino to Python through serial communication and plot the acquired data on Python IDE. The integrated development environment (IDE) of an Arduino has a serial monitor which will be used for real time simulation of the system. XCTU which is the program for the configuration of XBee wireless technology will be used to configure the communication settings between two XBee modules. The collected data will also be displayed by means of 20X4 LCD module and a program built in Python script.

Table 1.1: Methods of displaying data collected from the sensors

Type of Sensor	**Method of Data Analysis**
Humidity Sensor	Python script/LCD
Soil Moisture Sensor	Python Script/LCD
CO_2 Sensor	Python Script/LCD
Temperature Sensor	Python Script/LCD
Light Sensor	Python Script/LCD

1.8 Data sources

Collected data will be temperature, humidity, soil moisture, carbon monoxide (CO) and light intensity. This will be done by means of sensors which cause will actuation depending on the nature of the signal. This data will be sent to a spreadsheet and kept as a record of events for the greenhouse. The records can be used to calibrate the system for best results in future plant growth and harvesting.

1.9 Data collection techniques

In this study, an exertion has been made to outline and build up an Arduino based information lumberjack (data logger) for greenhouse. This work will concentrate on climatic parameters, such as, temperature, humidity, carbon monoxide (CO) content and light intensity. The data lumberjack (logger), Arduino situated in ATmega328 and the Python program will all from part of the complete system. The Arduino microcontroller board is used which has inbuilt ADC and other periphery equipment critical for operation. The physical parameter will be detected by the sensors and will be changed into analog signal. This analog signal will then be taken to the Arduino board ADC pins which will then be changed over into a proportionate digital signal. The digital signal of microcontroller will be shown on the LCD or dumped in the database or if not the information will be sent to the PC through the USB serial port. The serial port information will then be transported in Python program for calculation and graphical display.

1.10 Reliability issues and validity

The system's reliability to give accurate results may depend on the choice of sensors used to acquire data. Some sensors are quick to respond to physical conditions than others. Also in the design procedure calculations have to be made depending on the controlled devices. The program for the microcontroller also has to be the best and must be designed to fully operate all the controlled devices. Over the past years, greenhouse control systems have been designed and data acquired successfully using sensors. This research aims at improving the methods of automation surely with improved results of plant growth for monitored greenhouse. GSM systems so far have been designed which are partially automated as human intervention is involved by means SMS messages. Also, the aim of this research focuses on the designing of a fully automated system capable of acquiring and storing data for better plant growth and harvesting with the aid of WSNs and fuzzy logic control.

1.11. Dissertation outline

Chapter 2 will focus on the literature review of different technologies used in micro-climatic greenhouse monitoring system.

Chapter 3 will carry out the methodology used in this study. The focus will be to match the research questions, objectives and the design and experimental procedures.

Chapter 4 will present and analyze our results as per our methodology in chapter 3

Chapter 5 will conclude the study and make any future recommendations thereof.

CHAPTER 2: Wireless technologies for smart agriculture

2.1. Introduction

The advancement in sensing and communication technologies has significantly brought down the cost of deployment and running of a feasible precision agriculture sensor enabled networks. Emerging wireless technologies with low power needs and low data rate capabilities have been developed which perfectly suit precision agriculture [1]. The sensing and communication can now be done on a real-time basis leading to better response times. The wireless sensors are cheap enough for wide spread deployment and offer robust communication through redundant propagation paths. For this research study, it is important to first do a survey or literature review of existing wireless technologies and find out about their hardware platforms in order to understand which types and protocols were used.

2.2 Wireless sensor networks in greenhouse monitoring

Remote sensor networks that are developed by usage of XBee remote innovation embed on a ZigBee protocol which is a standard communication for low-power, wireless mesh networking. XBee is a brand of radio that support an assortment of communication protocols, including ZigBee, 802.15.4, and Wi-Fi, among others [11]. ZigBee systems can associate together in a few distinct designs or topologies to give the system its network structure.

The major ZigBee topologies are:

i. Pair: The most straightforward system is unified with only two radios, or hubs. One hub must be a facilitator (coordinator) with the goal that the system can be framed. The other can be designed as a switch (router) or an end gadget (end point).
ii. Star: This system course of action is likewise genuinely basic. An organizer (coordinator) radio sits at the focal point of the star topology and associates with a hover of end gadgets. Each message in the framework must go through the facilitator radio, which courses them as required between gadgets. The end points don't speak with each other directly.
iii. Mesh: The work design utilizes switch hubs (router) in addition to the coordinator radio. These radios can pass messages along to different routers and end points as required. The coordinator (a unique type of switch) coordinates the transfer of messages among different end points. Different end points might be connected to any router or to the coordinator.

Endpoints are capable of creating and getting data with the assistance of their parent nodes. The coordinated function of such nodes form what is known as a topology.

iv. Cluster tree: This is a system design where routers shape a spine of sorts, with end points bunched around every router. It's not altogether different from a mesh configuration.

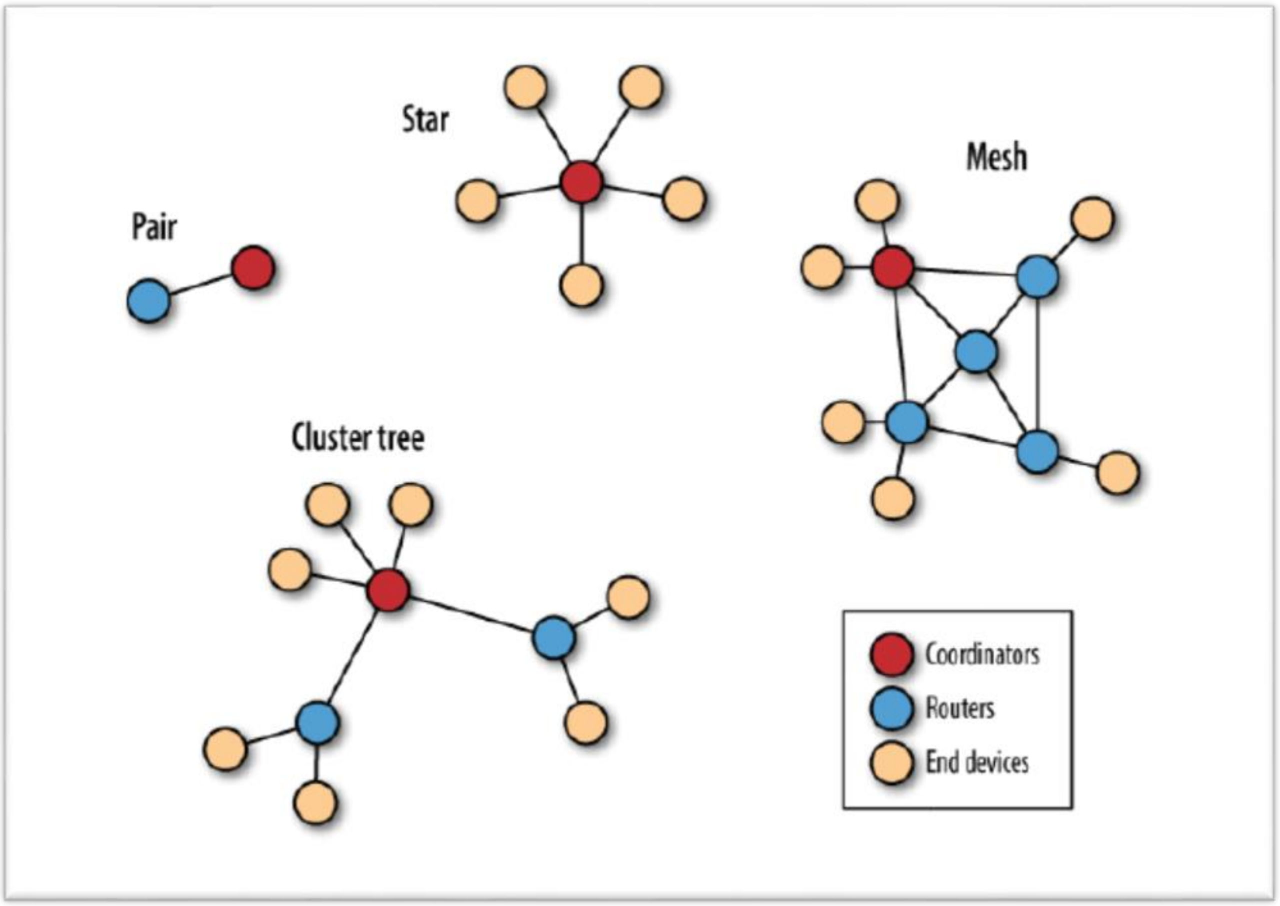


Figure 2.1: ZigBee Pair, Star, Mesh, and Cluster tree topologies

2.3. Common Hardware Platform for most wireless sensor networks

There are several different hardware platforms used in the sensor technology. Basically, the microcontroller board is the best remote system at this stage. The perfect small scale controller boards are the ones that have locally available ADC change, for example, Arduino board microcontrollers. The Arduino board usually have high sampling rate and solid high data acquisition.

Notwithstanding the high sampling rate and solid high-determination data acquisition, a standout feature among the hardware board is its little size and the capacity to work without external power supply [12]. Figure 2.2. shows an outline of ViFDAQ demonstrating the microcontroller and firmware center.

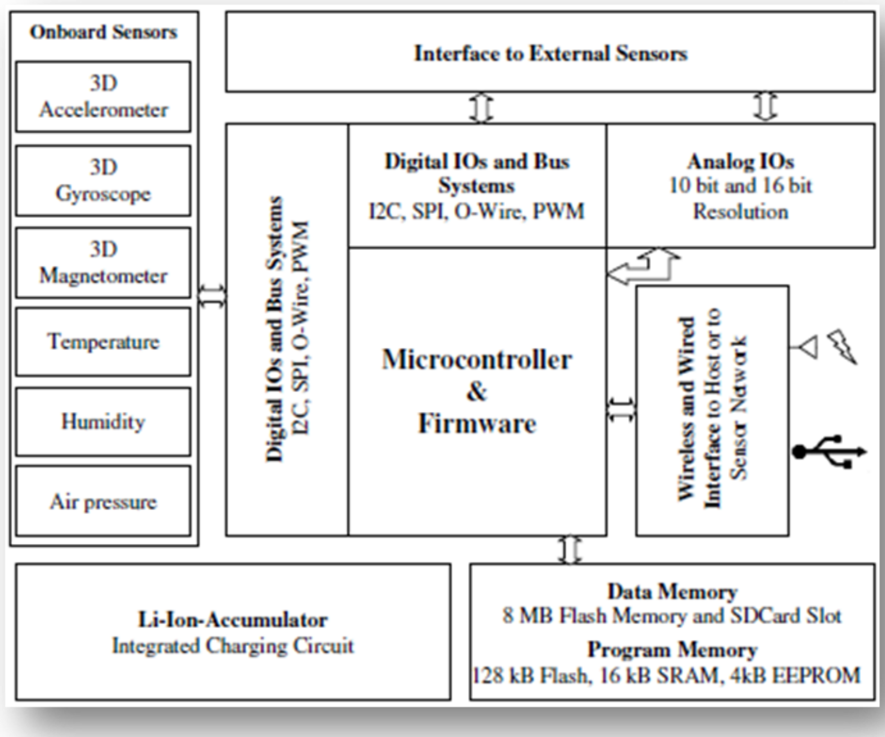

Onboard Sensors
3D Accelerometer
3D Gyroscope
3D Magnetometer
Temperature
Humidity
Air pressure
Interface to External Sensors
Digital IOs and Bus Systems
I2C, SPI, O-Wire, PWM
Digital IOs and Bus Systems
I2C, SPI, O-Wire, PWM
Analog IOs
10 bit and 16 bit Resolution
Microcontroller & Firmware
Wireless and Wired Interface to Host or to Sensor Network
Li-Ion-Accumulator
Integrated Charging Circuit
Data Memory
8 MB Flash Memory and SDCard Slot
Program Memory
128 kB Flash, 16 kB SRAM, 4kB EEPROM

Figure 2.2: Functional piece outline of ViFDAQ demonstrating the microcontroller and firmware center, analog and digital IOs, installed information and program memory, locally available sensors, and interfaces to outer sensors, host or sensor networks [12]

Information extraction in sensor networks can be accomplished in a wide range of routes depending on the application in use. Usually, this can be improved through an adaptation of the system, an estimation of the route through a graphical user interface (GUI) [12]. With this, a user can measure a task, based on his programming abilities and knowledge of the system. Programming skills are necessary for a user to program the platform. Figure 2.3. shows the flowchart for programming a measuring task for the ViFDAQ sensor platform.

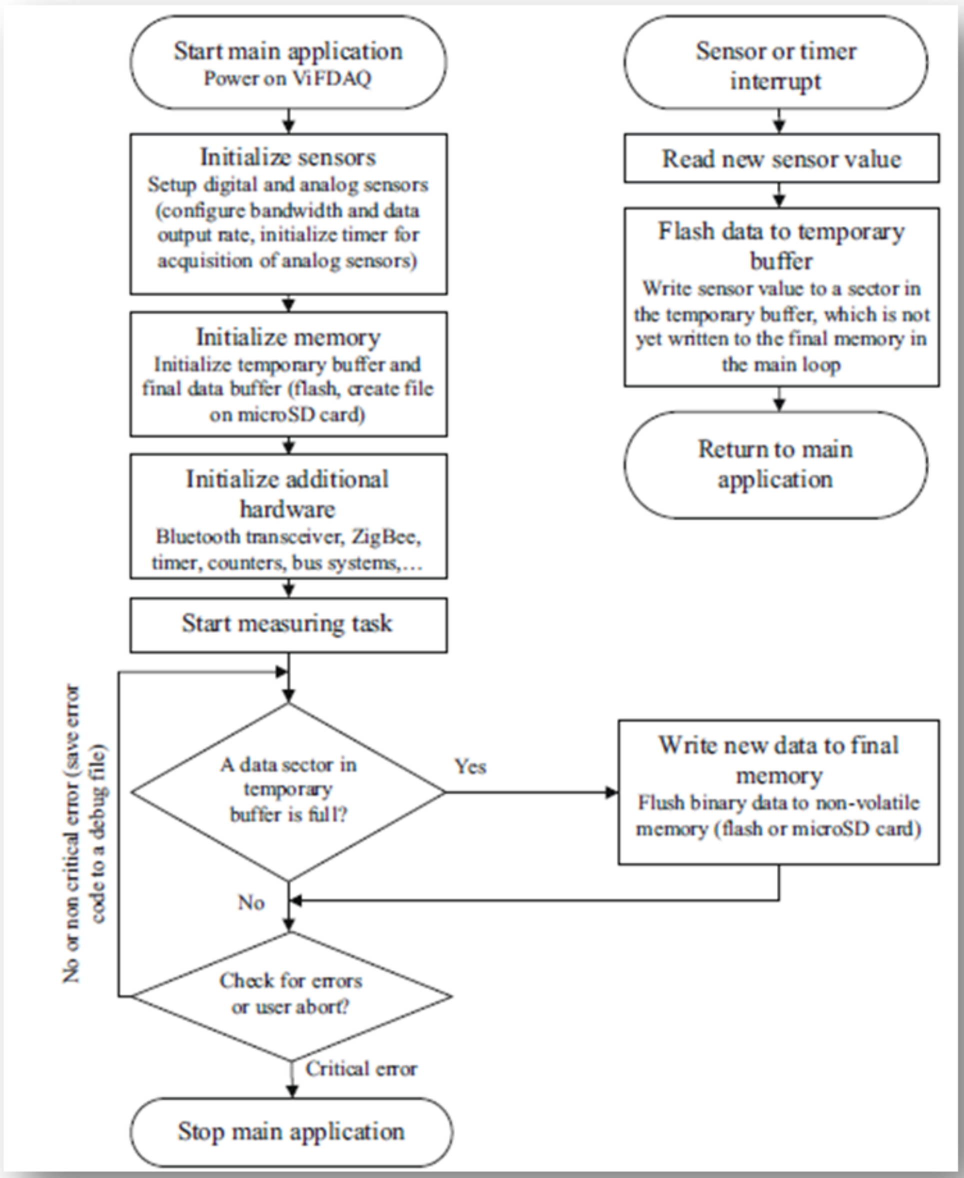

Start main application
Power on ViFDAQ
Initialize sensors
Setup digital and analog sensors (configure bandwidth and data output rate, initialize timer for acquisition of analog sensors)
Initialize memory
Initialize temporary buffer and final data buffer (flash, create file on microSD card)
Initialize additional hardware
Bluetooth transceiver, ZigBee, timer, counters, bus systems,…
Start measuring task
A data sector in temporary buffer is full?
Yes
No
Write new data to final memory
Flush binary data to non-volatile memory (flash or microSD card)
Check for errors or user abort?
No or non critical error (save error code to a debug file)
Critical error
Stop main application
Sensor or timer interrupt
Read new sensor value
Flash data to temporary buffer
Write sensor value to a sector in the temporary buffer, which is not yet written to the final memory in the main loop
Return to main application

Figure 2.3: Flowchart for programming a measuring task for the ViFDAQ sensor platform [12]

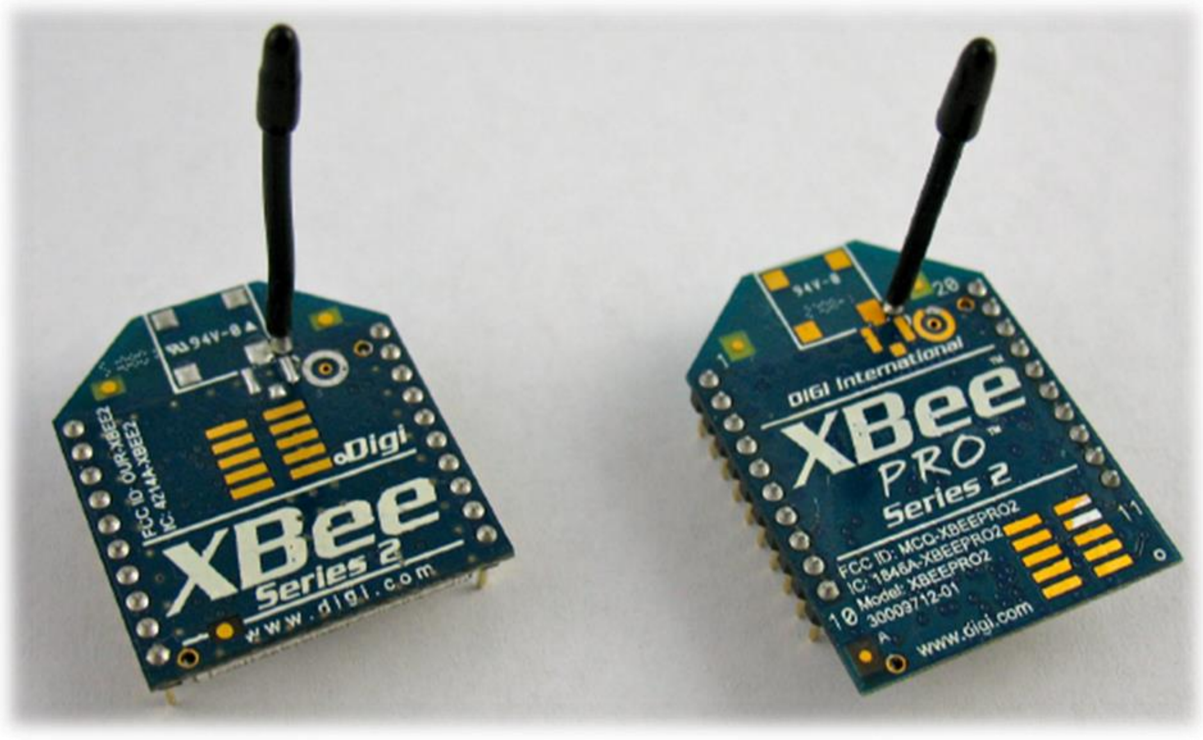


Figure 2.4: XBee radios: Regular and Pro types

2.4 Raspberry Pi platform

Raspberry Pi is a microcomputer board that has useful information/yield pins (GPIO). It uses Linux commands or language that controls its on board activities. Raspberry Pi is utilized as an installed Linux board which is outlined in view of the ARM V8 microcontroller architecture. The board has an Ethernet interface which runs the information web server. Usha et. al. [13] utilized Raspberry Pi to program and control a water pump and to monitor the plant growth and development through a wireless fidelity (Wi-Fi) enabled webcam that was streaming the information on the condition of the farm on PC. For all intents and purposes, the Raspberry Pi, in the case of Usha et. al. [13], was the center of the entire sensor network. A computerized water system framework was created to improve water use for horticultural yields. The objective was to control the water pump automatically, monitor the plant development using the Wi-Fi enabled webcam. Also the whole process of live streaming of the farm conditions were monitored live on Wi-Fi enabled android phones [13]. It must be stated that Raspberry Pi 3 can be configured into an XBee platform as shown in Figure 2.5.

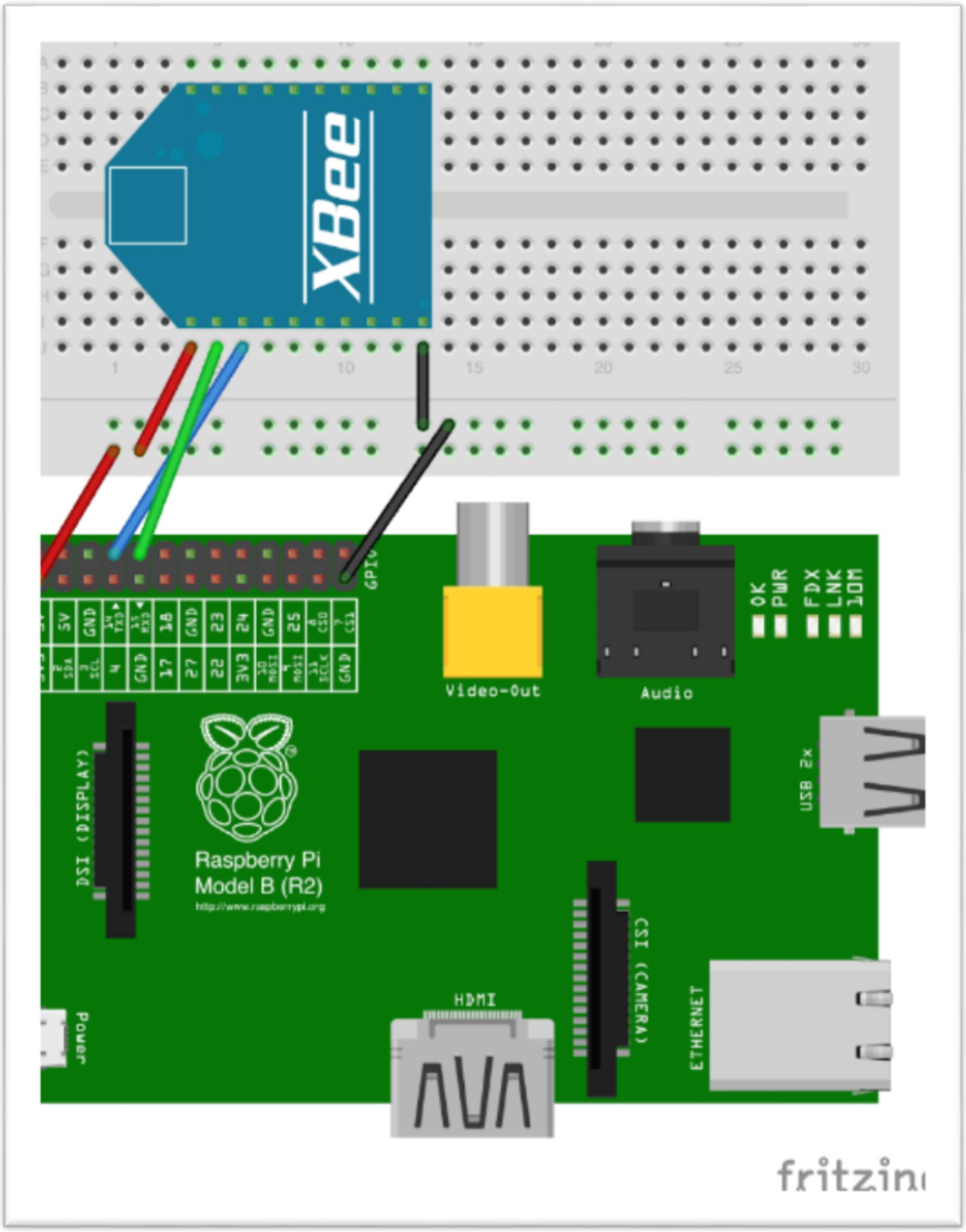


Figure 2.5: Raspberry Pi configured for XBee wireless radio

2.5Arduino microcontroller platform

The Arduino microcontroller board can be modified so as to function as an adapter for XBee radios [1] (see Figure 2.6.). This hack is so valuable, if the user does not have any desire to purchase an adapter or the user does not have a normal XBee adapter setup.

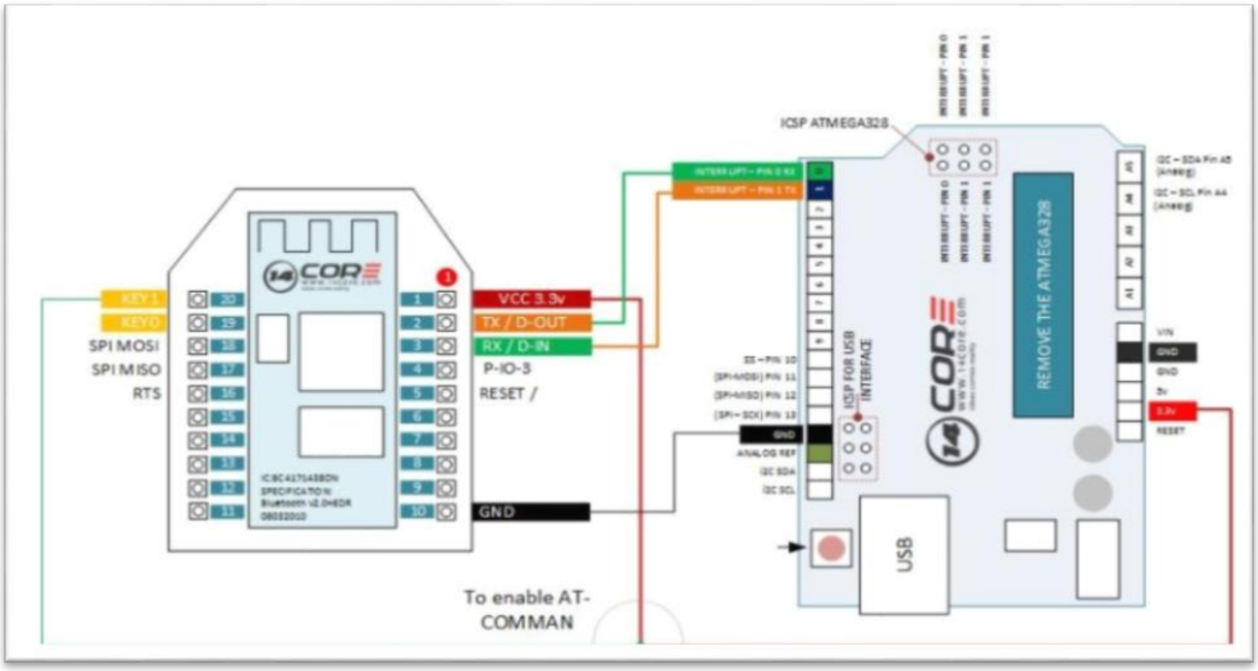


Figure 2.6: Arduino adapter hack for XBee wireless radio

2.6 Types of wireless sensor monitoring gadgets

There are a few contraptions available that are fit for checking the atmosphere conditions in the greenhouse. This section looks at two existing monitoring devices that were used for this research.

2.6.1 Win land Enviro Alert

Win land Enviro Alert (see Figure 2.7) has the ability of connecting up to four wired sensors and four remote sensors. Each detecting unit has its own particular assigned transfer yield, implying that the user can have transmitters initiate, dialers call or enact cautions when the modified limits are surpassed. This system has a remote scope of 300 meters between the sensor units and base units, making it perfect for monitoring long range applications.

The accompanying are its principle features [14]:

i. Data logging - catch and download sensor readings, alert and occasion history (has USB port for simple information recovery).
ii. Simultaneous operation of eight sensors.
iii. Easy to read expansive LCD display.
iv. Auxiliary yield transfer for nearby capable of being heard alert.
v. Auxiliary alert quiet component with 10-minute clock.
vi. Temperature and humidity Hi/Lo settings.
vii. Output transfers can be arranged to be at first invigorated or de-empowered.
viii. Tamper evidence secret word bolt setting.
ix. Snap fit mounting base with primary lodging for simple access.
x. Plug in terminal strip connectors for less demanding wire end.
xi. Current sensor readings are shown on home screen.
xii. Sensor design held in memory if control is lost.

Following are the shortcomings that are with this system:

i. Sensing units are separately sold.
ii. Is not intended for nursery atmosphere checking.
iii. High control utilization.
iv. Doesn't support remote mesh network.

Figure 2.7: Win land Enviro Alert

2.6.2 Watchdog Wireless Crop Monitor

Guard dog Wireless Crop Monitor (see Figure 2.8) can be utilized as a part of or any sort of natural observing applications that require steady temperature and humidity checking [15]. It utilizes remote detecting units to gauge and remotely transmit the temperature and humidity readings. The base unit can bolster up to a most extreme of 16 detecting units and is able to drive at the same time each of the 16 detecting units anyplace inside 300 meters run. Be that as it may, if more than 16 detecting units are to be sent inside 300 meters of the base unit, extra checking unit is required. The base unit accompanies a misery alert and cautioning light that will enact when the modified edges are surpassed.

The following are its main features [15]:

i. Simultaneous operation of up to 16 sensing units.
ii. LCD display that is easy to read.
iii. Auxiliary output relay for a local audible alarm.
iv. Hi / Lo set points for temperature & humidity.
v. Output relays can be configured to be initially energized or de-energized.
vi. System is expandable (This means have multiple Watchdog Sensors operating on the same system network).

The following are the shortcomings of the system:

i. Sensors are separately sold.
ii. Doesn't support remote mesh network.
iii. Is not particularly intended for greenhouse climate monitoring but rather for general purpose environment monitoring.
iv. Very difficult to configure the system.
v. The system has no climate control capability.

Figure 2.8: Watchdog Wireless Crop Monitor

2.7 Routing protocols for wireless sensor networks

A routing protocol determines how routers speak with each other, scattering data that empowers them to choose courses between any two hubs on a network. Routing algorithms decide the particular decision of each route. Every router has some information on earlier routes appended to it. A routing protocol shares this data first among prompt neighbors, and after that through the system. Along these lines, routers pick up and learn of the topology of the system [16].

Despite the fact that there are many sorts of routing protocols, three noteworthy classes are in boundless use on IP systems:

i. Interior gateway protocol sort 1 are protocols such as the open shortest path first (OSPF) and IS-IS.
ii. Interior gateway protocol sort 2, separate vector steering conventions, for example, Routing Information Protocol, RIPv2, IGRP.
iii. Exterior passage protocols are routing protocols utilized on the Internet for exchanging routing data between autonomous systems, for example, Border Gateway Protocol (BGP) and Path Vector Routing Protocol (PVRP).

Wireless sensor systems can be classified into WSN protocol layers which make remote correspondence feasible [17]:

Physical Layer: The physical layer performs balance (modulation) on active signals and demodulation on approaching signals. It transmits data and gets data from a source.

Media Access Control (MAC) Layer: The elements of the MAC layer are to get to the system by utilizing carrier sense medium access with collision avoidance(CSMA/CA), to transmit signals for synchronization, and to give dependable transmission

Network Layer: The network layer is the layer that provides packet forwarding and establishes routing protocols amongst nodes or routers that are close.

Application Support Sublayer (APS): The application support sublayer (APS) gives the important application objects (endpoints) and the ZigBee device object (ZDO) to interface with the system layer for information and administrations.

2.8 ZigBee Protocol Architecture

Figure 2.9 demonstrates or rather shows the ZigBee layer stack. The ZigBee alliance built up the ZigBee device object (ZDO), the application support sublayer (APS), the network layer, and security administration layer. IEEE 802.15.4 is utilized for the MAC layer and physical layer [17].

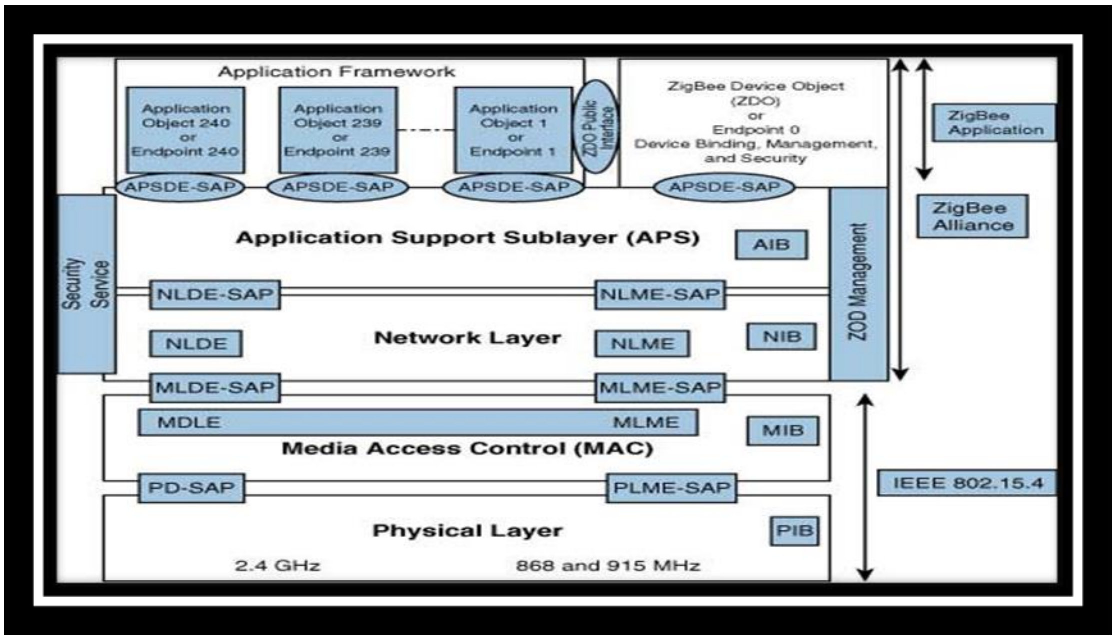


Figure 2. 9: ZigBee protocol architecture

The ZigBee convention design is partitioned into three areas, as follows [17]:

i. IEEE 802.15.4, which comprises of the MAC and physical layers.
ii. ZigBee layers, which comprise of the network layer, the ZigBee device object (ZDO), the application sublayer, and security administration.
iii. Manufacture application: Manufacturers of ZigBee devices can utilize the ZigBee application profile or build up their own application profile. EEE 802.15.4 indicates the physical layer and data link layer protocol for low-rate remote individual region systems. Be that as it may, this particular just concerns interchanges between gadgets that are inside each other's transmission range. For bigger sensor networks, the help of network layer protocols is required [18].

2.9 Power management in WSNs

Needs for power management in WSNs [19]:

i. Sensor nodes are battery driven so for this reason they must have a lifetime on the order of months to years.
ii. With thousands of physically embedded nodes, battery replacement is not an option.
iii. These networks may in some cases be required to solely operate on energy scavenged from the environment through seismic, photovoltaic or thermal conversion.

Micro controller units usually support various operating modes for power management. These modes include active, idle and sleep modes [19]. However, transitioning between operating modes involves a power and latency overhead. Radios can operate in four distinct modes of operation. These modes are Transmit, Receive, Idle, and Sleep. An important observation in the case of most radios is that operating in idle mode results in significantly high power, almost equal to the power consumed in the receive mode. Passive sensors in WSNs such as temperature consume negligible power relative to other components of the sensor node [19]. The operation of batteries in sensor nodes depend on many factors like battery dimensions and type of electrode material used to mention a few. Denser distributions of sensors lead to increasingly tracking results but it reduces network life time. It is desirable to avoid routes through regions of the network that are running low on energy resources, thus preserving them for future [19].

2.10 Related work

In the section were present some of the related work to our study. In literature a lot of studies have been done regarding the automation of agriculture or more precisely smart agriculture. To start with, Yang et al [20] designed an automatic surveillance system that was built upon a wireless sensor network. This system was used for ecological and natural environment surveillance. By equipping each wireless sensor node with an automated counting and trapping device, this system was capable of automatically reporting environmental conditions (e.g. Temperature, humidity, illumination, wind speed, wind direction, and rainfall volume) and the number of the Oriental fruit flies in real-time [20]. The user can access the sensed data via Internet and find the results of statistical analyses [20].

The humidity in a greenhouse has to be measured and controlled in its entire interior. This needs a distributed sensing network whose number of nodes depends on the greenhouse area. Other researchers such as Postolache et al [21], designed a system for humidity monitoring in green houses. Using this system, the humidity in a greenhouse was measured and controlled in its entire interior. When monitoring in this way there is need to have a distributed sensing network whose number of nodes depends on the greenhouse area [21].

Wang et al, [22] designed wireless mid-Infrared spectroscopy sensor network for automatic carbon dioxide fertilization in a greenhouse environment. This system was used to monitor and control the carbon dioxide in the green house. The control was based on mesh network of sensor nodes that were connected together and transmitted data to a single point where it was processed. In order to ensure accurate carbon dioxide injection, the CO_2 sensor was designed and implemented based on infrared technologies with waterproof membranes [22]. Basically the injection was done in four phases. Firstly, the CO_2 concentration sensor node based on nondispersive infrared (NDIR) technology with anti-condensation prevention and wireless communication was used to measure the required environmental factors. Secondly, the established wireless sensor network was able to read the values of the transmitted data. Thirdly, the interactive software platform based on MATLAB was actually used to process the data and to achieve synchronization with the remote terminals. Finally, after all of the above the control node would inject a suitable amount of CO_2 in the greenhouse following the transmitted control parameters by the software platform on the terminal. These four sections establish a closed-loop monitoring and control system to realize carbon dioxide measurement and fertilization

[22]. Lieschnegg et. al. [23], designed an autonomous sensor platform for environmental monitoring applications. The main aim of the research was to design sensor platform, which was versatile, easy to operate and with a measurement system that was capable of recording data for various tasks simultaneously and synchronizing the input of multiple sensors connected [23]. Postolache et. al. [21] designed a system for humidity monitoring in green houses. Using the system, the humidity in a greenhouse was measured and controlled by a group of connected sensor nodes. This needed a distributed sensing network whose number of nodes depended on the greenhouse area [21]. Postolache et. al. [21] designed this system such that the communication between the nodes and a host personal computer (PC) that was used for advanced data processing and data logging was performed through an acquisition and communication module.

To avoid cabling, the wireless network was used and the communication protocol that was used was based on blue tooth (class 1) and ZigBee [21]. Regarding software implementation, a serial port profile was used in order to emulate serial cable connections using RFCOMM between two peer devices [21].

A commonly practiced control scheme in current automated irrigation systems is called scheduling. That is, irrigation system can be scheduled to apply water for a predetermined period of time. The schedule can be implemented using timers connected to a number of sensor nodes. The scheduling method usually causes a significant amount of leachate to occur. In order to avoid this a method or a way of controlling leachate has to be activated. The most efficient way is to activate irrigation only when soil moisture is below a certain threshold. How accurate the schedule is, depends on skills and the interpretation of results by those in charge. The difficulty in the application of traditional feedback control in irrigation system lies in the nonlinear response and time delay of the soil moisture sensor (so called soil sensor continuum) [2]. If that happens, it is usually necessary to predict the soil moisture from a significantly delayed, nonlinear sensor’s response; otherwise, the plants will experience water shortage due to the sensor’s slow response. Researchers such as Tilt et al [2] designed an irrigation control system using fuzzy logic. The system used fuzzy logic in a way that resembled a human operator. It determined when the irrigation should begin based on the information from the soil moisture sensors. Additionally, the systems also learned the irrigation schedules by itself during the on-line control process.

Other such as Zualkernan et al [24], designed a ZigBee-based irrigation system for home gardens which was a wireless irrigation system for a smart home garden that can be integrated with existing smart home control systems. The system consisted of slave nodes and a master station each of which was equipped with a wireless microcontroller. Each slave node was equipped with a temperature sensor, a soil-moister sensor, a water valve, a microcontroller and a ZigBee transceiver. The slave microcontroller node reads the surrounding temperature of the garden's grass and trees along with soil moisture. Then, the frame is forwarded to the master station through a ZigBee ad-hoc network [24].

Mashhad et al [5], designed a fuzzy logic enabled dryer system whose control parameters were based on temperature, humidity and weight. When starting the device, the product's weight was measured by a weight sensor. According to the product type that was selected manually by the user, drying operation was either started or abandoned. If the drying process was started the heater was turned on and the drying process was initiated, with gradually increase in temperature if need arose. There was an alternation between the amount of moisture and the temperature needed. For example, if the temperature was too, high the moisture content was turned on. This alternate and proper balance between temperature and moisture would make the fruit either not to be too hard and not too soft. The desired weight of the product was reached through this process. The desired product's weight together with the drying operation was performed with less power and fuzzification [5]. Finally, Salazar et al [25], developed, through fuzzy logic a system that monitored the temperature of thermal plasma torch. In this work, the environment of Arduino interaction was a water refrigeration system with a thermal plasma torch. The Arduino board received an electrical signal from a temperature sensor and acted on by controlling a frequency inverter.

2.11 Chapter summary

In this chapter we looked at the topologies of a ZigBee protocol and discussed each in detail. The covered topologies were pair, star, mesh and cluster tree. WSN common hardware protocol for micro-controllers such Arduino was dealt with and explained in detail. Power management for wireless sensor networks was explained to be by means sensor nodes that are battery driven and for this reason, must have a lifetime on the order of months to years. Routing protocols for wireless sensor networks were also explained in detail with the advantages and disadvantages of each also looked at. Finally, the previous works of other researchers were reviewed.

CHAPTER 3: Methodology

3.1. Introduction

The research method used in this study is quantitative research since experiments covered explain how the system is built and tested. The results of these experiments will be used to draw up results and conclusions. The goal of this research study was to develop a system that makes use of fuzzy logic decisions to monitor temperature, humidity, carbon monoxide gas, light intensity, and soil moisture content without intervention of human beings in a greenhouse environment. Abad and Mashhad [5] built a system based on fuzzy logic to control temperature and humidity of a dryer plant. Fuzzy control has also been used by Singh, Jha and Nandwana [26] to design a solar powered fuzzy controlled irrigation system for cultivation of green vegetable plants in India.

The experiments covered in this research study were as follows:

i. Experiment 1: Measuring soil moisture content using grove moisture sensor.
ii. Experiment 2: Measuring temperature and humidity using DHT11 sensor.
iii. Experiment 3: Measuring Carbon monoxide content using MQ7 gas sensor.
iv. Experiment 4: Measuring light intensity using LDR.
v. Experiment 5: Demonstrate fuzzy logic control of temperature, humidity, moisture content, Carbon monoxide concentration and light intensity.

The system's wireless radio modules used in this research study were XBee radios which were configured as three different stations shown in Figure 3.1. The system's block diagram consists of sensor station with XBee radio configured as a ZigBee router in the XCTU program, a second XBee radio of the coordinator station configured as a ZigBee coordinator in the XCTU program and the third XBee radio in the central station also configured as a ZigBee router. Each station consisted of an Arduino microcontroller which was used to perform control algorithms. The Arduino microcontrollers that were available in all three stations were connected to XBee radios via Arduino wireless proto-shields. The Arduino wireless proto-shield could be converted to USB through a slide switch which had to be set to "Serial" when XBee was to be programmed. After successfully programming the XBee the tiny slide switch of the wireless proto-shield was set to "Micro" in order to achieve successful communication between the XBee and the Arduino microcontroller.

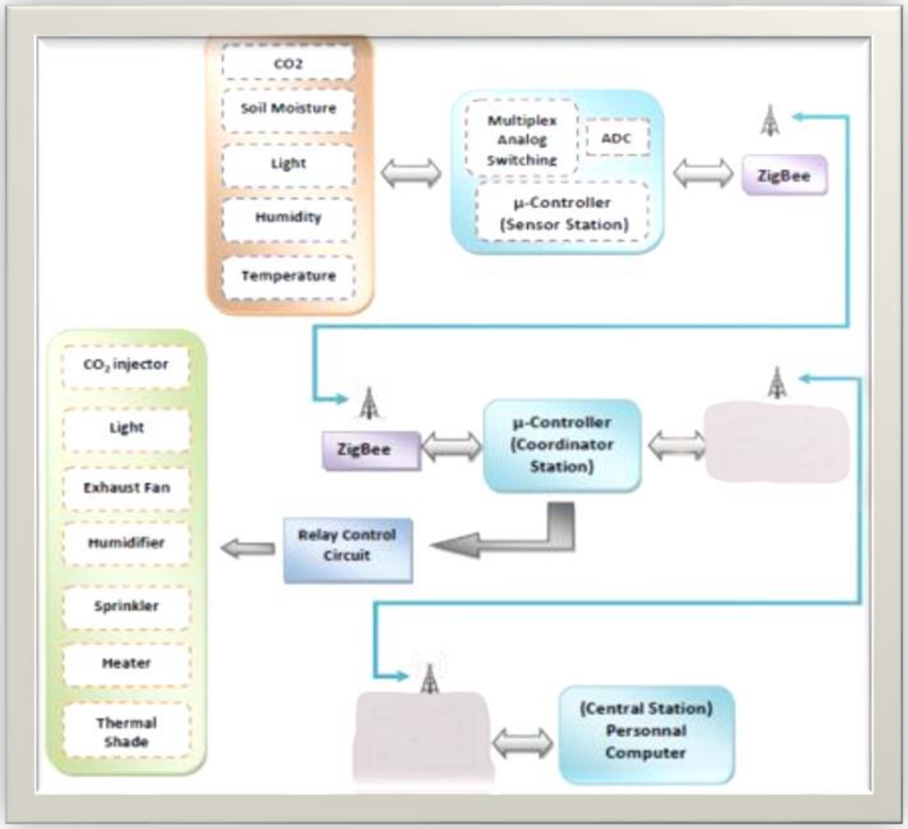


Figure 3.1: Block diagram of the entire system

3.2 Block diagrams for configuring XBee radios

Before looking at the entire system design we need to first understand how XBee radio modules were configured in the XCTU program so that they could be able to communicate with each other effectively. In this study, two of the XBee radios were configured as routers and one of them as a coordinator. XCTU program was used for configuring XBee radios in ZigBee protocol [27]. Figure 3.2 shows a block diagram of XBee configuration as a router while Figure 3.3 shows a block diagram of XBee configuration as a coordinator.

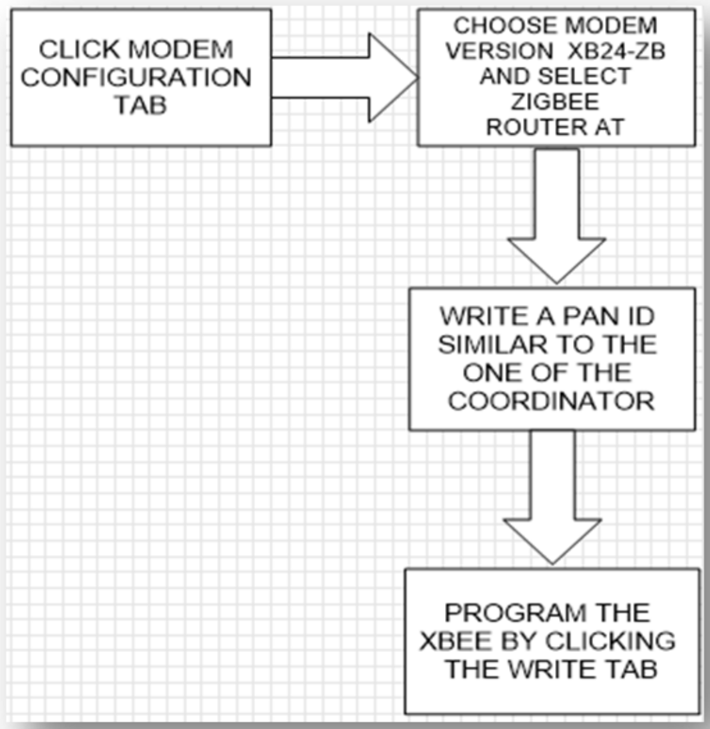

CLICK MODEM CONFIGURATION TAB
CHOOSE MODEM VERSION XB24-ZB AND SELECT ZIGBEE ROUTER AT
WRITE A PAN ID SIMILAR TO THE ONE OF THE COORDINATOR
PROGRAM THE XBEE BY CLICKING THE WRITE TAB

Figure 3.2: XBee router configuration

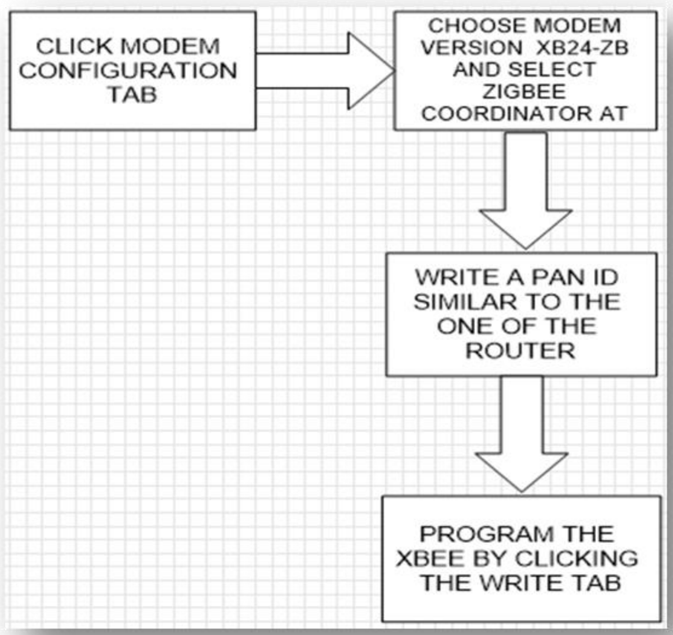
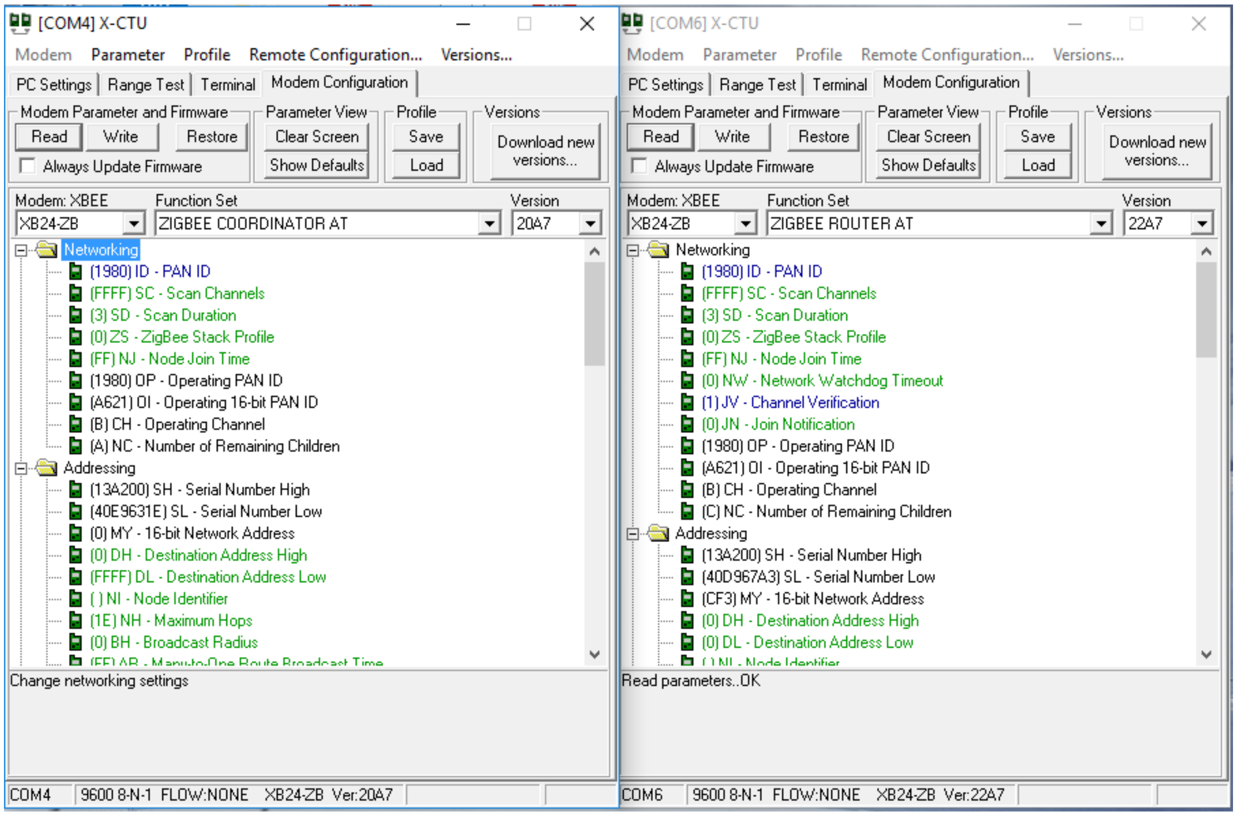

CLICK MODEM CONFIGURATION TAB
CHOOSE MODEM VERSION XB24-ZB AND SELECT ZIGBEE COORDINATOR AT
WRITE A PAN ID SIMILAR TO THE ONE OF THE ROUTER
PROGRAM THE XBEE BY CLICKING THE WRITE TAB

Figure 3.3: XBee coordinator configuration


[COM4] X-CTU
Modem Parameter Profile Remote Configuration... Versions...
PC Settings | Range Test | Terminal | Modem Configuration
Modem Parameter and Firmware: Read Write Restore
Always Update Firmware
Parameter View: Clear Screen Show Defaults
Profile: Save Load
Versions: Download new versions...
[COM6] X-CTU
Modem Parameter Profile Remote Configuration... Versions...
PC Settings | Range Test | Terminal | Modem Configuration
Modem Parameter and Firmware: Read Write Restore
Always Update Firmware
Parameter View: Clear Screen Show Defaults
Profile: Save Load
Versions: Download new versions...

```
Modem: XBEE     Function Set                 Version
XB24-ZB         ZIGBEE COORDINATOR AT        20A7
Networking
  (1980) ID - PAN ID
  (FFFF) SC - Scan Channels
  (3) SD - Scan Duration
  (0) ZS - ZigBee Stack Profile
  (FF) NJ - Node Join Time
  (1980) OP - Operating PAN ID
  (A621) OI - Operating 16-bit PAN ID
  (B) CH - Operating Channel
  (A) NC - Number of Remaining Children
Addressing
  (13A200) SH - Serial Number High
  (40E9631E) SL - Serial Number Low
  (0) MY - 16-bit Network Address
  (0) DH - Destination Address High
  (FFFF) DL - Destination Address Low
  ( ) NI - Node Identifier
  (1E) NH - Maximum Hops
  (0) BH - Broadcast Radius
  (FF) AR - Many-to-One Route Broadcast Time
Change networking settings
COM4   9600 8-N-1  FLOW:NONE   XB24-ZB  Ver:20A7
```

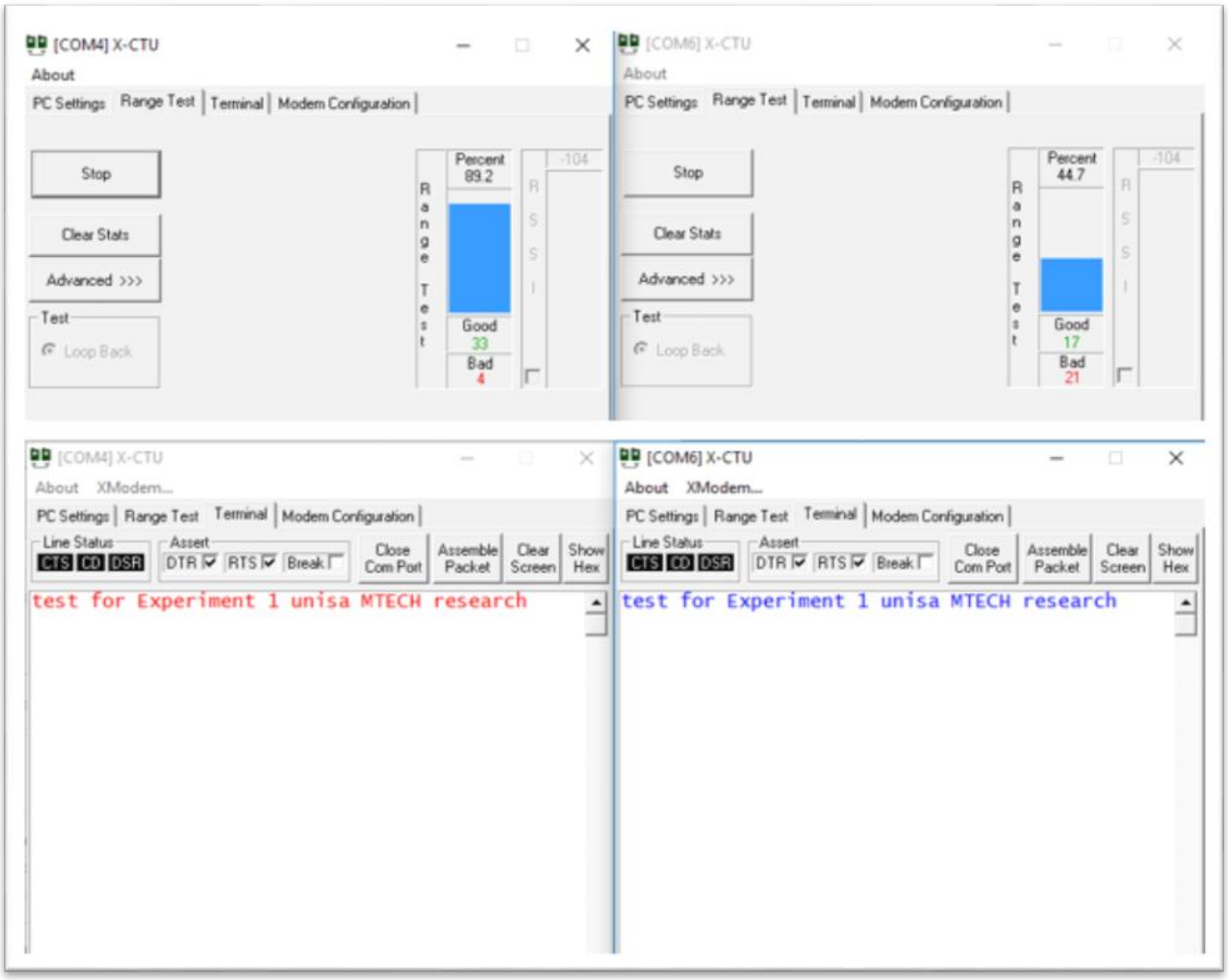
```
Modem: XBEE     Function Set                 Version
XB24-ZB         ZIGBEE ROUTER AT             22A7
Networking
  (1980) ID - PAN ID
  (FFFF) SC - Scan Channels
  (3) SD - Scan Duration
  (0) ZS - ZigBee Stack Profile
  (FF) NJ - Node Join Time
  (0) NW - Network Watchdog Timeout
  (1) JV - Channel Verification
  (0) JN - Join Notification
  (1980) OP - Operating PAN ID
  (A621) OI - Operating 16-bit PAN ID
  (B) CH - Operating Channel
  (C) NC - Number of Remaining Children
Addressing
  (13A200) SH - Serial Number High
  (40D967A3) SL - Serial Number Low
  (CF3) MY - 16-bit Network Address
  (0) DH - Destination Address High
  (0) DL - Destination Address Low
  ( ) NI - Node Identifier
Read parameters..OK
COM6   9600 8-N-1  FLOW:NONE   XB24-ZB  Ver:22A7
```

Figure 3.4: Configuration of XBee router and coordinator on XCTU platform

XCTU from DIGIKEY is an open source program available online and is used for the configuration and programming of XBee radio modules [28]. Figure 3.4 shows how XBee radio was configured as a router and coordinator using the XCTU program. Once the configuration was completed, the write tab was clicked to program the XBee radio. A read tab could be used to check configurations of a preprogrammed XBee radio when necessary. One of the important aspects when configuring XBee radios of the same network was to make sure that they had same PAN IDs. If the PAN IDs were different there would be no communication between the router configured XBee radios and the coordinator configured XBee. The destination addresses (DH and DL) were configured as shown in Figure 3.4.

Table 3.1: Router and Coordinator configurations in the XCTU program

	Router configurations	**Coordinator configurations**
1.Baud rate	9600	9600
2.Response time out	1000	1000
3.Modem	XB-24ZB	XB-24ZB
4.Version	22A7	20A7
5.Function set	ZigBee Router AT	ZigBee coordinator AT
6.Pan ID	1980	1980



```
[COM4] X-CTU — Range Test
Percent 89.2
Good 33
Bad 4

[COM6] X-CTU — Range Test
Percent 44.7
Good 17
Bad 21

[COM4] X-CTU — Terminal
test for Experiment 1 unisa MTECH research

[COM6] X-CTU — Terminal
test for Experiment 1 unisa MTECH research
```

Figure 3.5: Communication test between XBee radio modules configured as router and coordinator.

The router XBee was connected to comport 6 while the coordinator XBee was connected to comport 4 as shown in Figure 3.5. To perform the test for effective data transfer between the two XBee radios, text was written on the terminal window of the router XBee (comport 6). To show that the two XBee radios communicated successfully, the written text automatically appeared in red font on the terminal window of the coordinator XBee (comport 4) as shown in Figure 3.5.

3.3 Sensor station design

The sensor station was responsible for collecting data from the sensors and transmitted it to the coordinator station as seen in Figure 3.1. In this study, the sensor station was designed by embedding a wireless proto-shield (see Figure 3.6) into an Arduino Mega 2560 (see Figure 3.7). The XBee radio (see Figure 3.8) was configured as a router then connected to an Arduino through the wireless proto-shield.

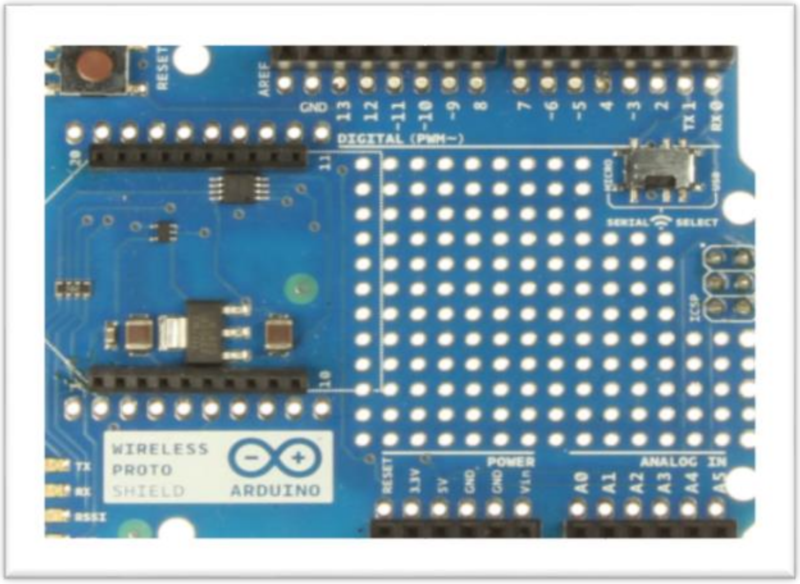


Figure 3.6: Wireless Proto shield

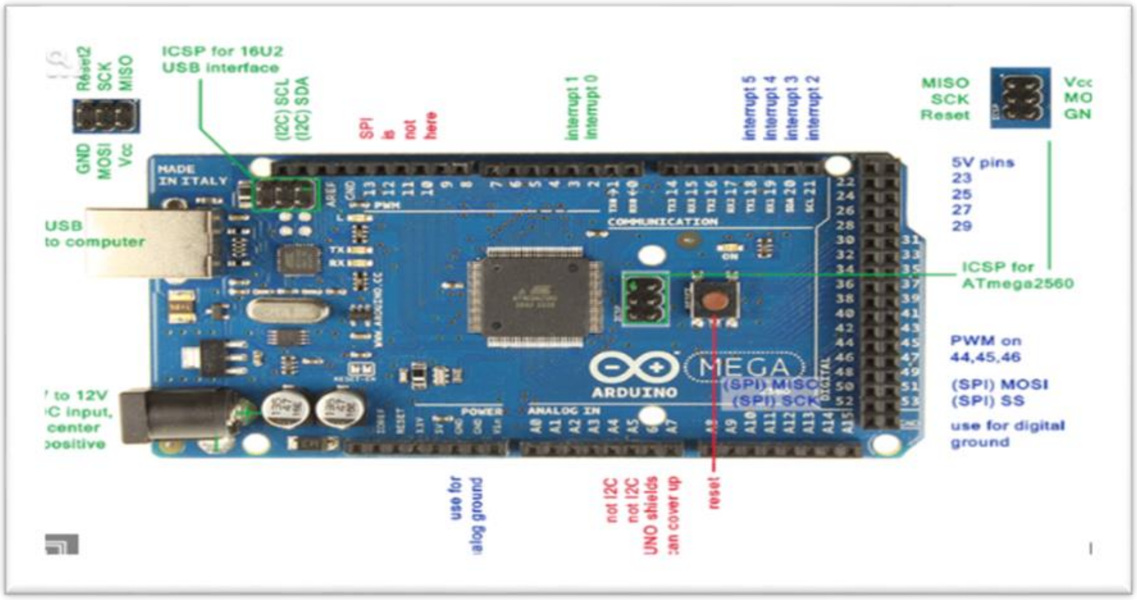


Figure 3.7: Arduino Mega 2560

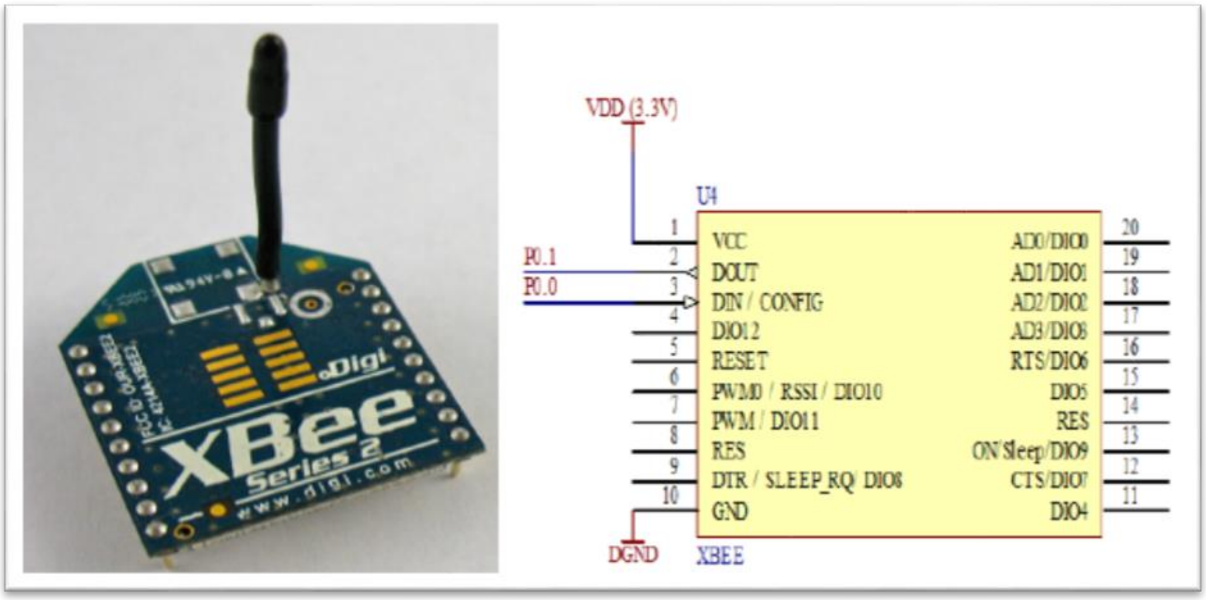


Figure 3.8: Series2 XBee radio with its connection layout

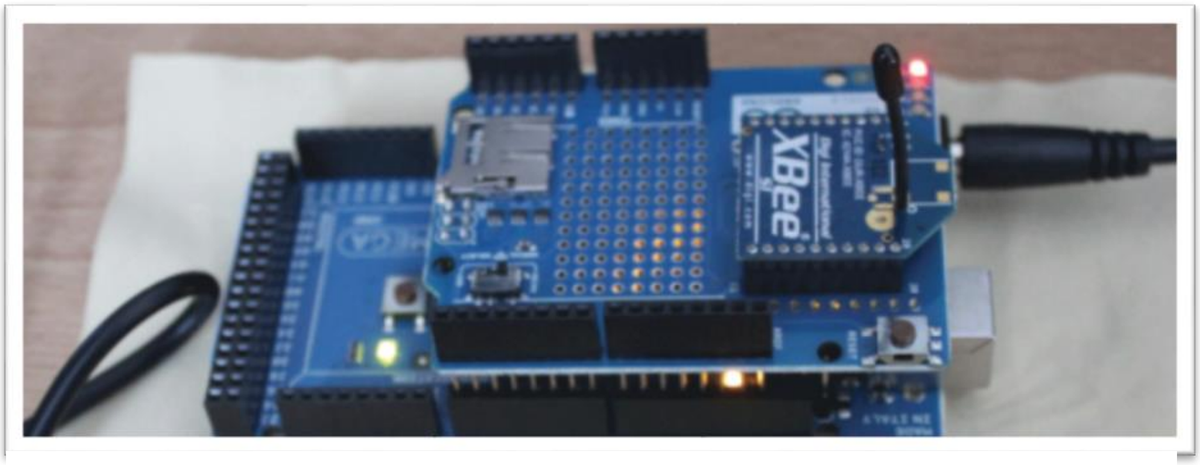
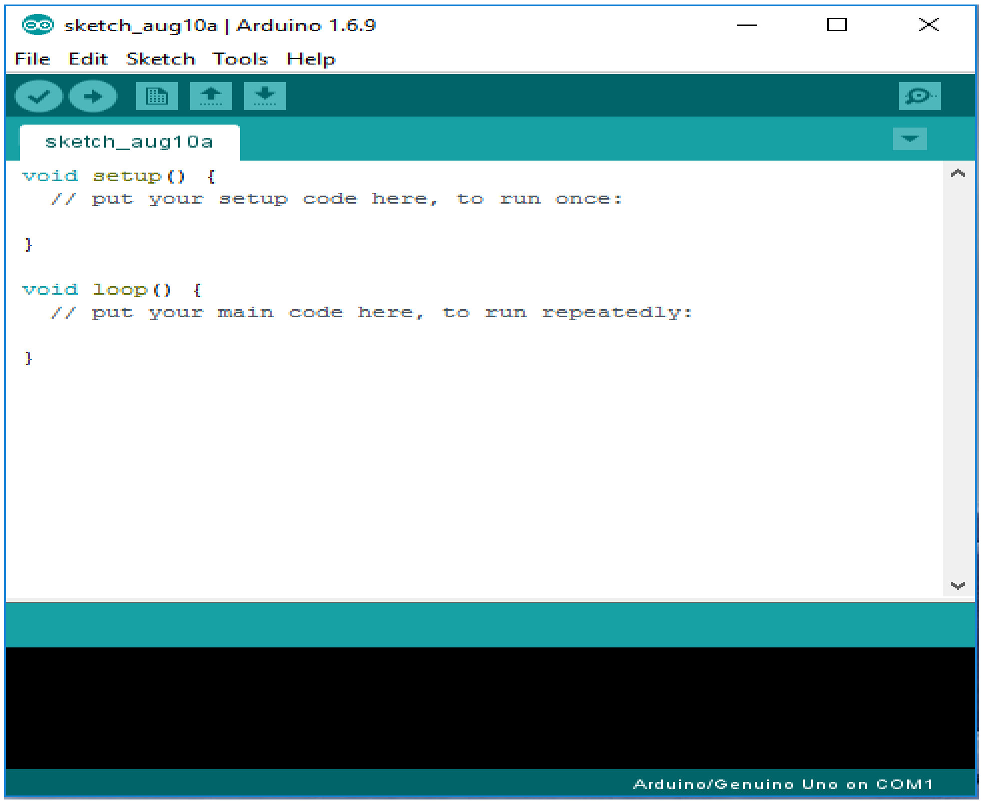


Figure 3.9: The Arduino with the XBee and Wireless Proto Shield installed

3.3.1 Putting it altogether for the sensor station

When programming the XBee, the tiny switch on top of the wireless shield was set to “Serial” and the Arduino program memory was emptied. An empty program memory was achieved by erasing the Arduino microcontroller in the Arduino IDE. In order to make the XBee communicate with the Arduino after the Arduino was programmed, the tiny switch on top of the wireless shield was set to “Micro”. Included in Appendix A is the Arduino code of the sensor station. The Arduino IDE is the environment for writing the Arduino code. Figure 3.10 shows the Arduino integrated development environment (IDE).


sketch_aug10a | Arduino 1.6.9
File Edit Sketch Tools Help
sketch_aug10a
Arduino/Genuino Uno on COM1

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 3.10: Arduino IDE

3.3.2 Sensor data analysis and complete circuit diagram of the sensor station

The sensors used in the sensor station were DHT11, MQ-7 carbon monoxide sensor, LDR sensor and Grove moisture sensor. These sensors were connected to the wireless proto-shield's Arduino inputs. Before looking at the connection of sensors in the sensor station, the sensors are first described.

i. Grove moisture sensor

Shown in Figure 3.11 is the grove moisture sensor. It was used to detect the degree of moisture content in the soil. In this study, it was used for activating the sprinklers in the coordinator station whenever the degree of the soil moisture was deemed dry by the Arduino microcontroller and cut off the sprinklers whenever the soil was humid to wet. These fuzzy decisions were made by fuzzy inference programmed in the Arduino microcontroller of the coordinator station after it had received data that was wirelessly transmitted from the sensor station.

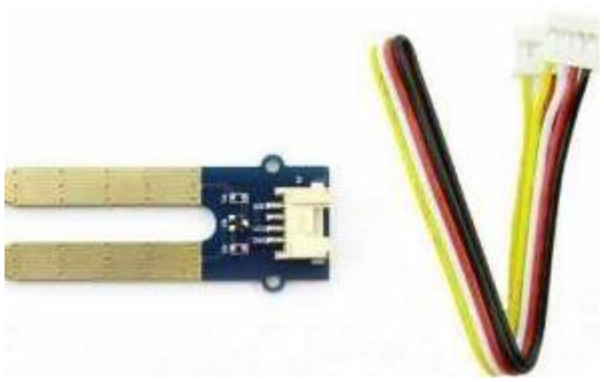


Figure 3.11: Grove moisture sensor

Specifications of Grove moisture sensor

Working voltage: 3.3~5V

Working current: 35mA

Sensor Output Value in dry soil: 0~ 300

Sensor Output Value in damp soil: 300~700

Sensor Output Value in water: 700 ~ 950

PCB measure: 2.0cm X 6.0cm

ii. DHT 11 Temperature and humidity sensor

The DHT11 is the dual-purpose sensor in the sense that it is used to measure both temperature and humidity. In this study, it was used by the sensor station to measure both humidity and temperature. The temperature and humidity data collected by the DHT11 sensor was transmitted wirelessly to the coordinator station where fuzzy decisions were made in order to control a heater, humidifier or fan. Figure 3.12 shows a DHT11 sensor.

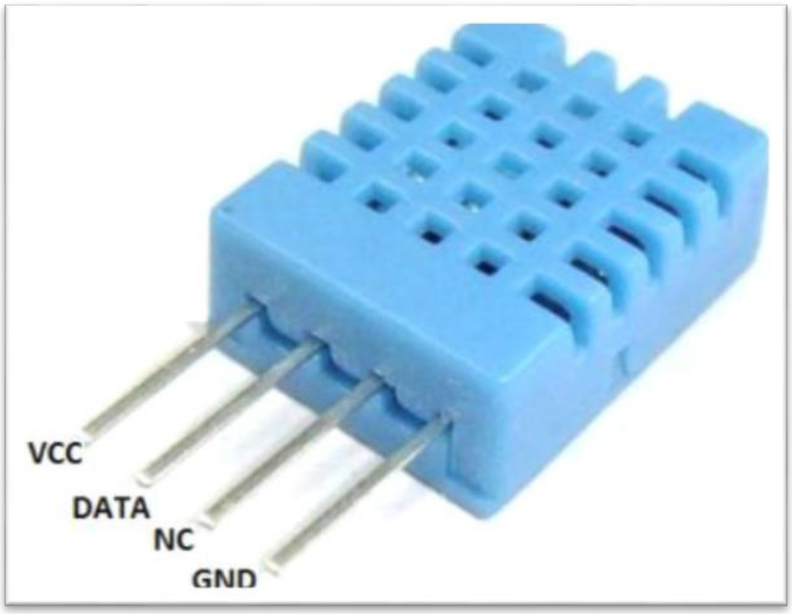


Figure 3.12: DHT11 Sensor

Table 3.2: Specifications of DHT11

Item	Measurement Range	Humidity Accuracy	Temperature Accuracy	Resolution	Package
DHT11	20-90%RH 0-50 ℃	±5%RH	±2℃	1	4 Pin Single Row

iii. LDR sensor

Since plants need enough light for photosynthesis [29], the LDR sensor was chosen as the perfect light sensor for detecting the amount of light intensity available in the green house. The LDR sensor data was transmitted wirelessly from the sensor station to the coordinator station where fuzzy decisions were made in order to control light intensity by switching the bulb on or off. Figure 3.13 shows an LDR sensor.

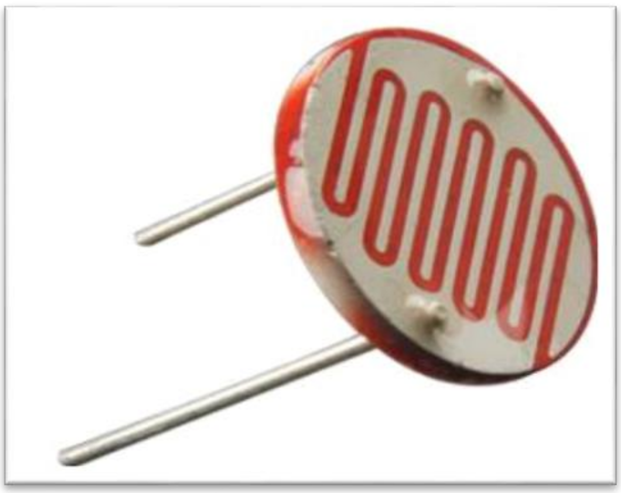


Figure 3.13: LDR sensor

Figure 3.14 shows the characteristics curve of the LDR sensor.

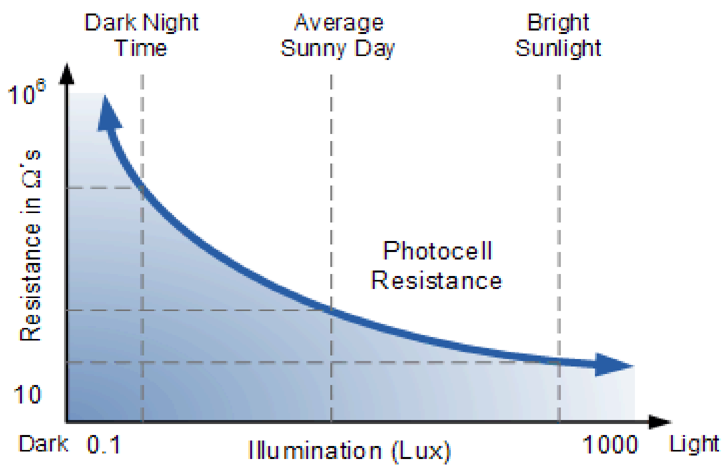


Figure 3.14: Characteristics curve of the LDR

iv. MQ7 carbon monoxide sensor

MQ7 is an easy to utilize Carbon Monoxide (CO) sensor and is reasonable for detecting CO concentrations noticeable all around. The MQ7 sensor can distinguish CO gas concentrations somewhere in the range of 20 to 2000ppm. This sensor has a high affectability and quick reaction time. The sensor's yield is a simple resistance.

The drive circuit is exceptionally straightforward in the sense that you should simply control the radiator loop with 5V, include a heap resistance, and associate the yield to an ADC. In this study, this sensor was made available at the sensor station and was used to measure the presence of carbon monoxide in the greenhouse. The collected data was transmitted wirelessly to the coordinator station. Since carbon monoxide is odorless gas that can destroy the plant roots, the collected data was used by the coordinator station to make fuzzy decisions in terms of whether carbon dioxide gas should be injected or not as a form of neutralizing CO gas. The injected carbon dioxide will be useful to plants for photosynthesis and root development [29]. Figure 3.15 shows an MQ7 gas sensor.

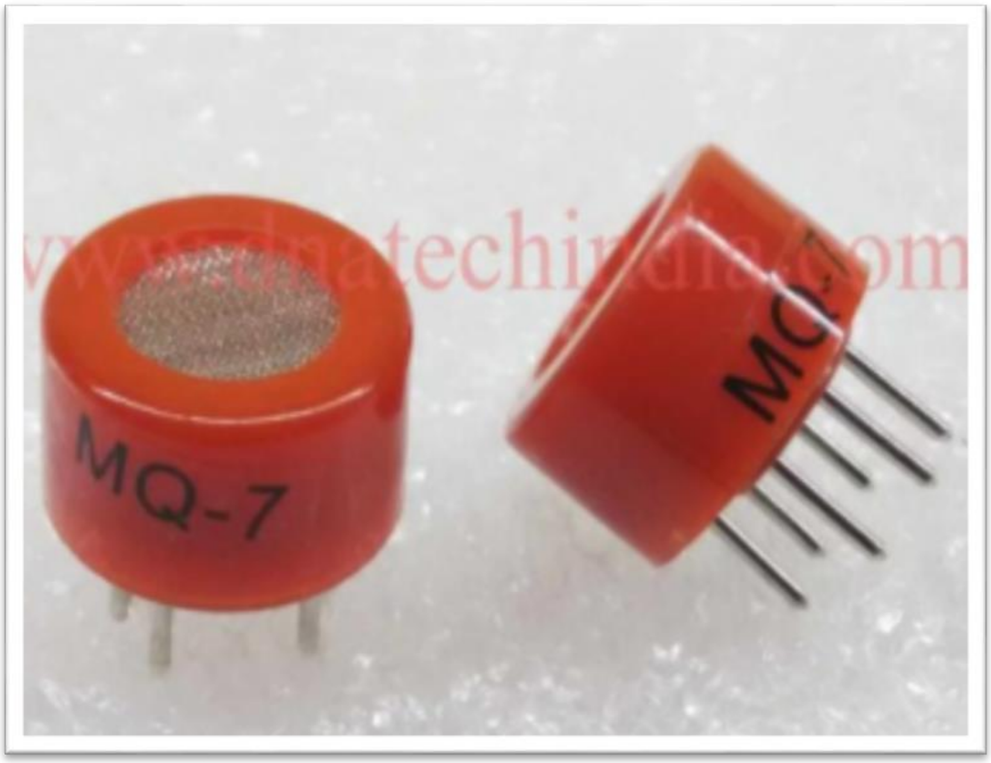
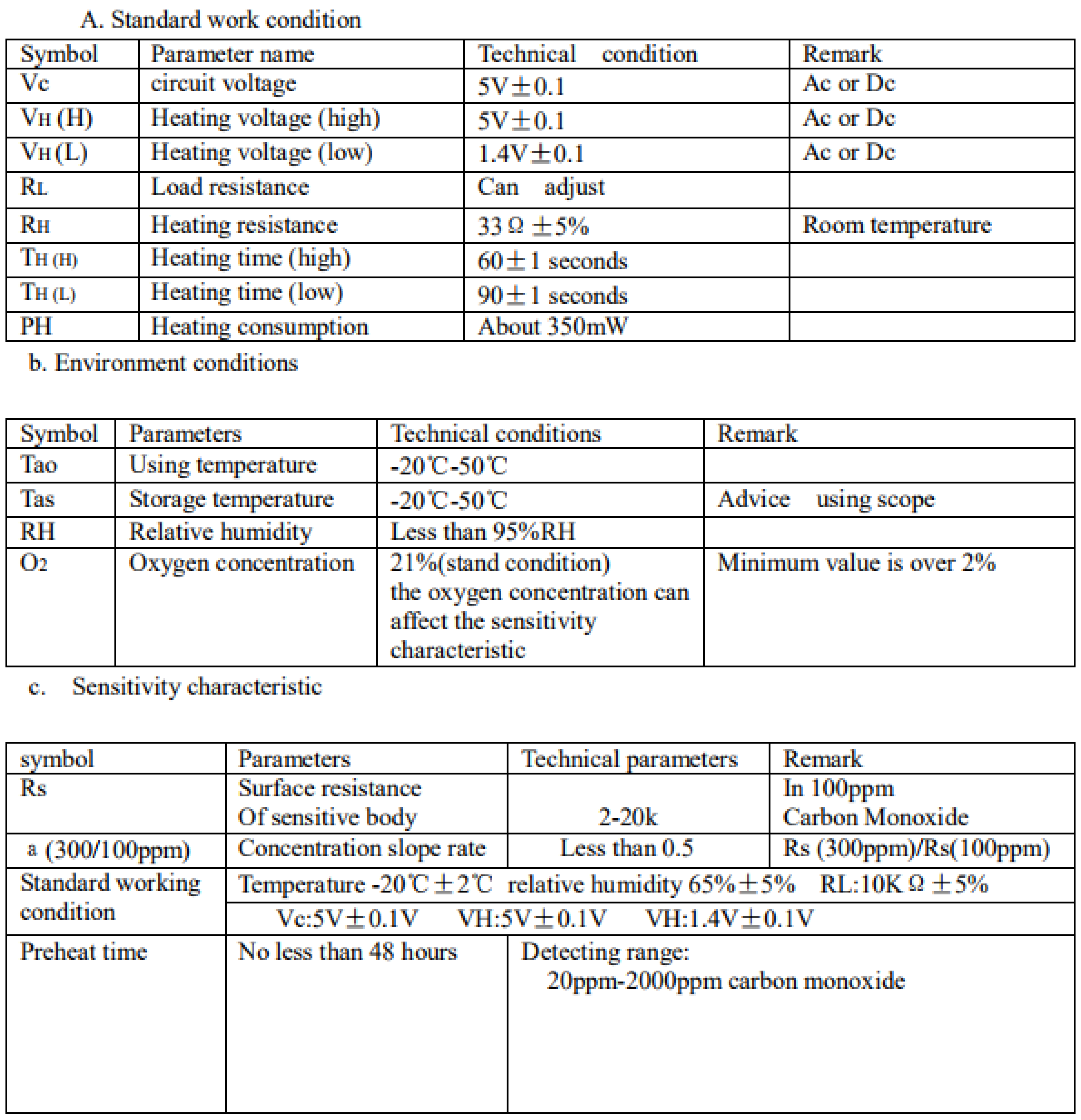


Figure 3.15: MQ7 gas sensor

Table 3.3: Specifications of MQ-7 Carbon monoxide sensor



A. Standard work condition

Symbol	Parameter name	Technical condition	Remark
Vc	circuit voltage	5V±0.1	Ac or Dc
V_H (H)	Heating voltage (high)	5V±0.1	Ac or Dc
V_H (L)	Heating voltage (low)	1.4V±0.1	Ac or Dc
R_L	Load resistance	Can adjust	
R_H	Heating resistance	33Ω ±5%	Room temperature
$T_{H\,(H)}$	Heating time (high)	60±1 seconds	
$T_{H\,(L)}$	Heating time (low)	90±1 seconds	
PH	Heating consumption	About 350mW	

b. Environment conditions

Symbol	Parameters	Technical conditions	Remark
Tao	Using temperature	-20℃-50℃	
Tas	Storage temperature	-20℃-50℃	Advice using scope
RH	Relative humidity	Less than 95%RH	
O_2	Oxygen concentration	21%(stand condition) the oxygen concentration can affect the sensitivity characteristic	Minimum value is over 2%

c. Sensitivity characteristic

symbol	Parameters	Technical parameters	Remark
Rs	Surface resistance Of sensitive body	2-20k	In 100ppm Carbon Monoxide
a (300/100ppm)	Concentration slope rate	Less than 0.5	Rs (300ppm)/Rs(100ppm)
Standard working condition	Temperature -20℃±2℃ relative humidity 65%±5% RL:10KΩ±5%		
	Vc:5V±0.1V VH:5V±0.1V VH:1.4V±0.1V		
Preheat time	No less than 48 hours	Detecting range: 20ppm-2000ppm carbon monoxide	

Shown by Figure 3.16 is the sensitivity characteristics curve of the MQ-7 carbon monoxide sensor.

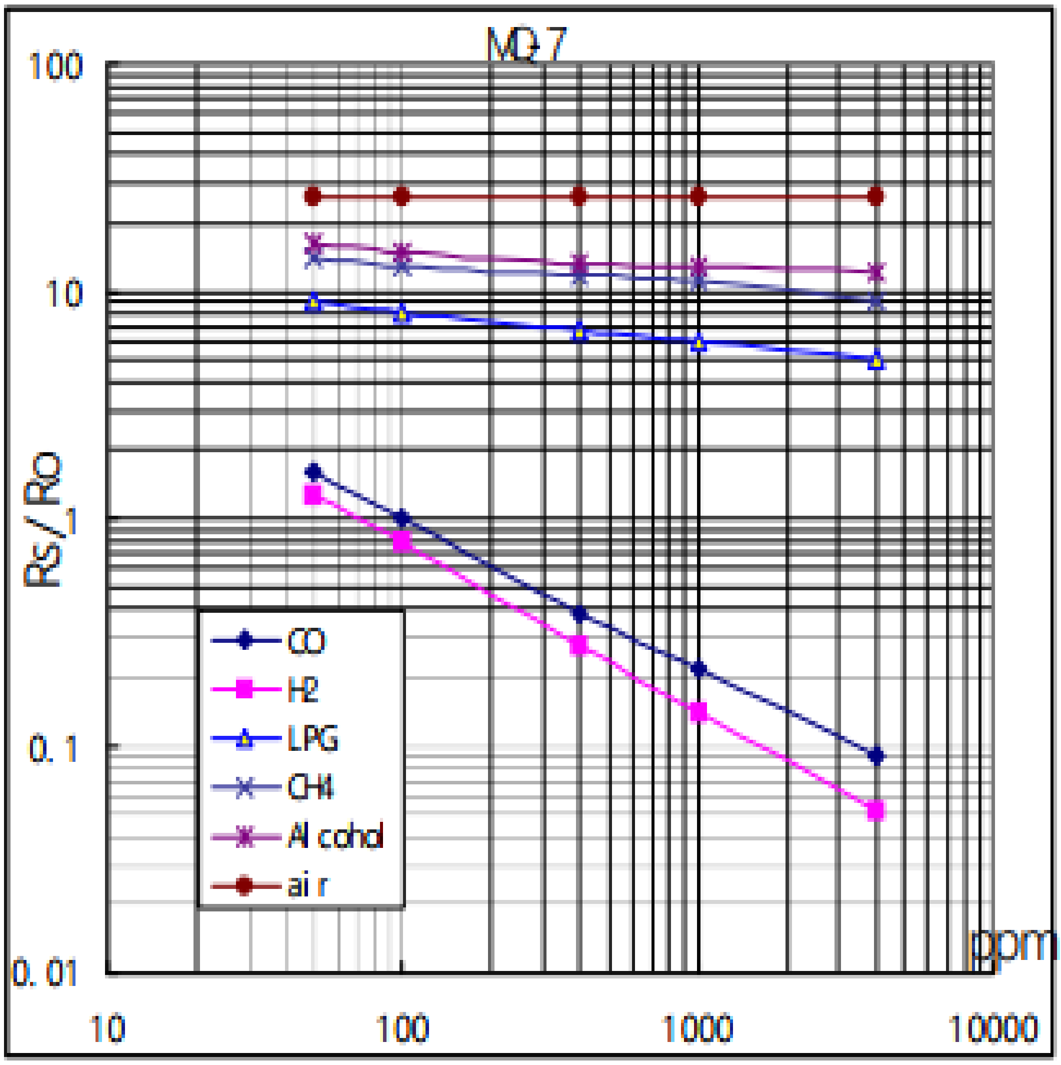


Figure 3.16: Sensitivity characteristics curve of the MQ-7

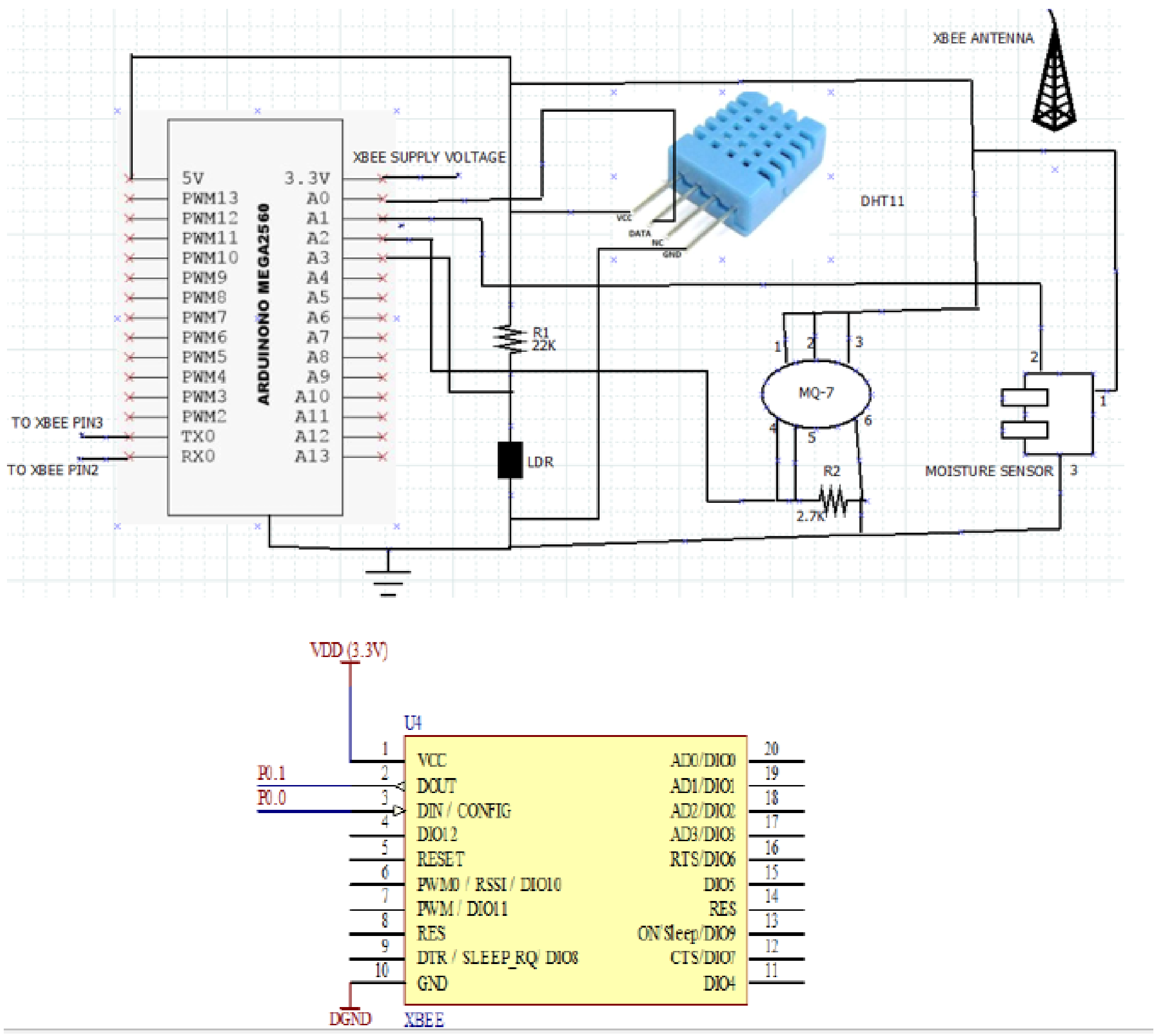


Figure 3.17: Complete circuit diagram of the sensor station

The data collected by the sensors was transmitted wirelessly by the XBee radio to the coordinator station. The coordinator station used the received data from the sensor station to monitor the environment of the greenhouse by controlling a heater, fan, humidifier and a light bulb. Figure 3.18 shows the sensor station designed for this research study.

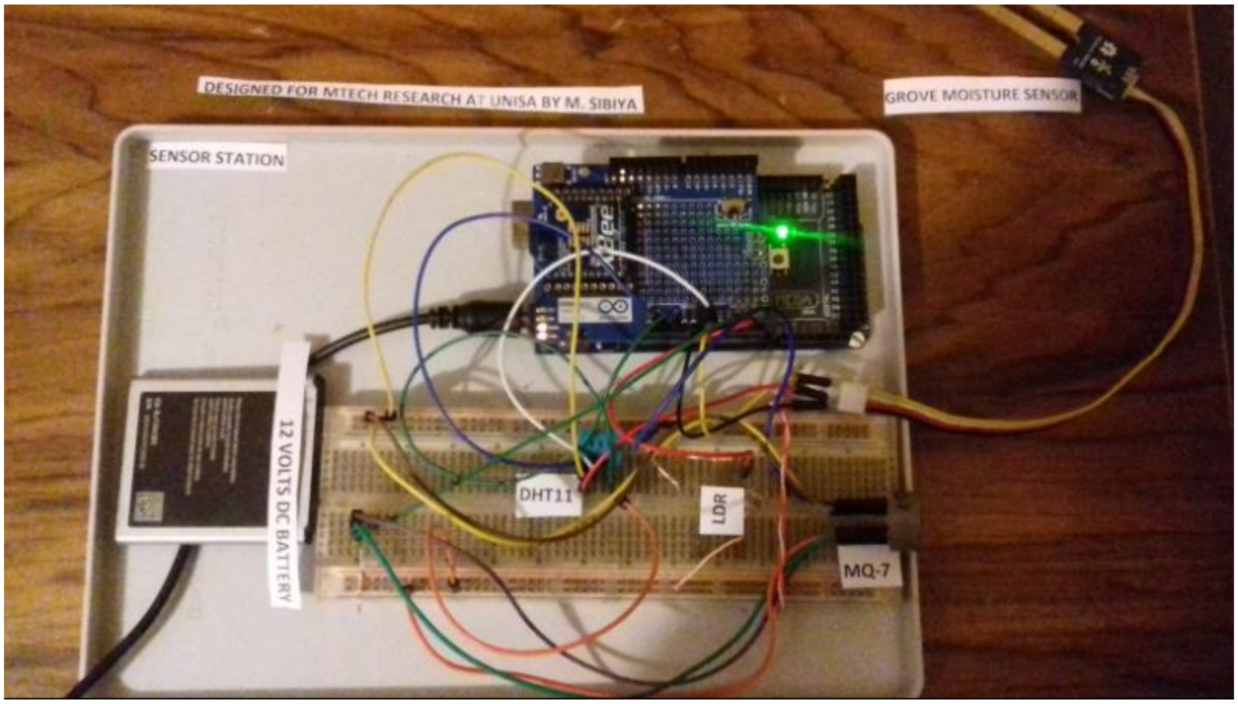


Figure 3.18: Designed sensor station

3.3 Coordinator station design

The coordinator station was designed the same way as the sensor station. The wireless proto-shield was connected to the Arduino mega 2560 and then on top of the wireless proto-shield an XBee was connected. This blend of connections is shown in Figure 3.9. The additional components were 20x4 LCD, 8 relay-module, LED and a Bluetooth module. The android smart phone was used to change the LCD screen display whenever the user needed to display gas data. Included in Appendix B is the Arduino microcontroller code for the coordinator station.

Table 3.4: Coordinator station components

Component/Device	**Description**
LED	Red
Bluetooth module	HC-05
LCD	20 x4 character display
Relays	8-relay module
Samsung smart phone	J3 model

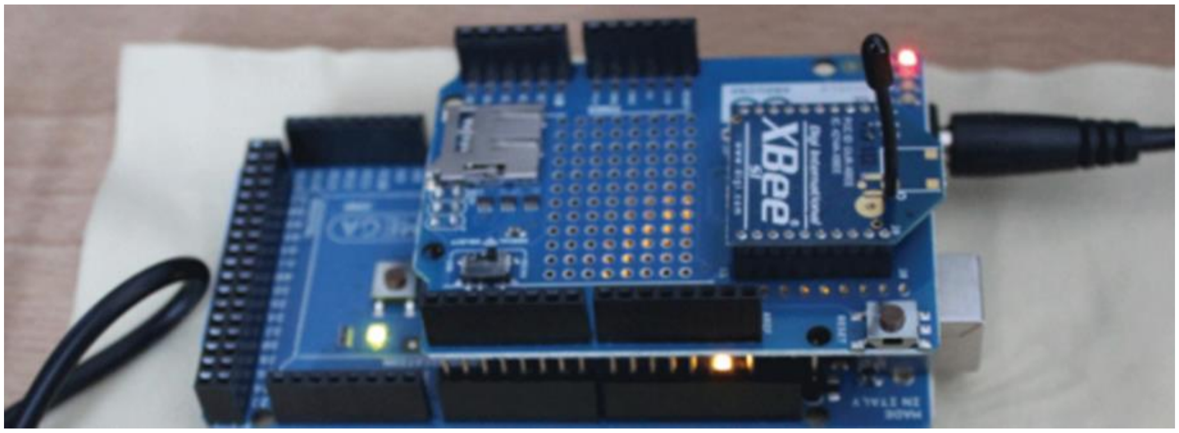


Figure 3. 19: The Arduino with XBee and wireless proto-shield installed

3.3.1. LCD and Bluetooth module functions in the coordinator station

The LCD was used to display data collected from the sensor station. However, when the coordinator station was reset, the LCD displayed the system's title as shown in Figure 3.20. This information was displayed for five seconds and then disappeared. After this time delay, data collected by sensors in the sensor station was displayed on the LCD as shown in Figure 3.21.

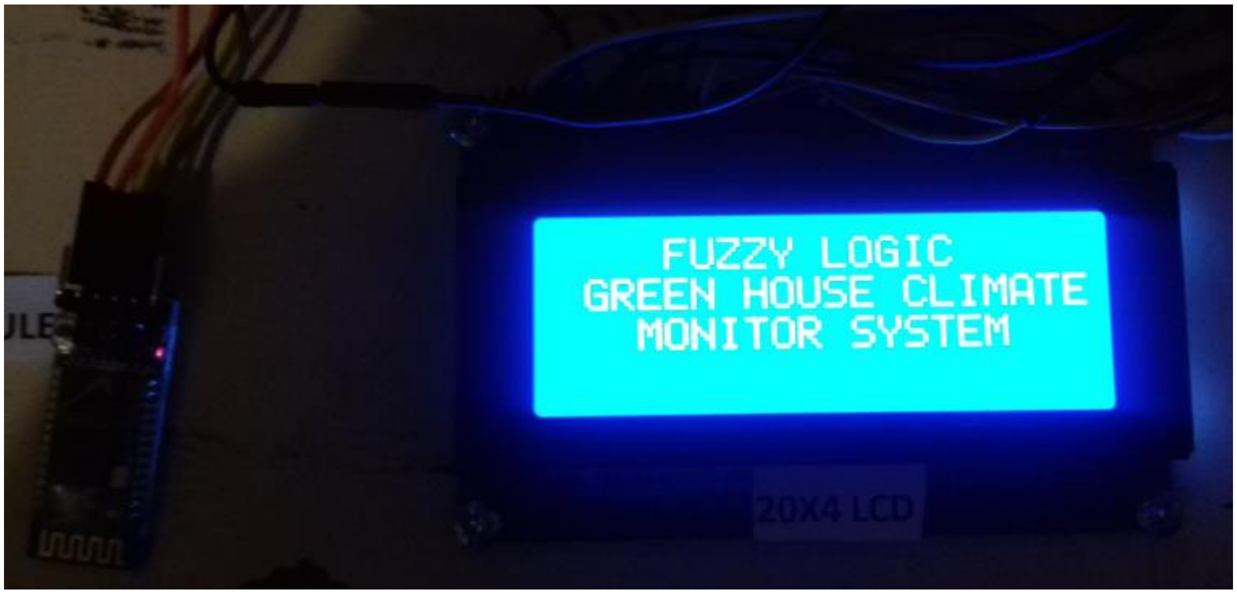


Figure 3.20: LCD display at power reset

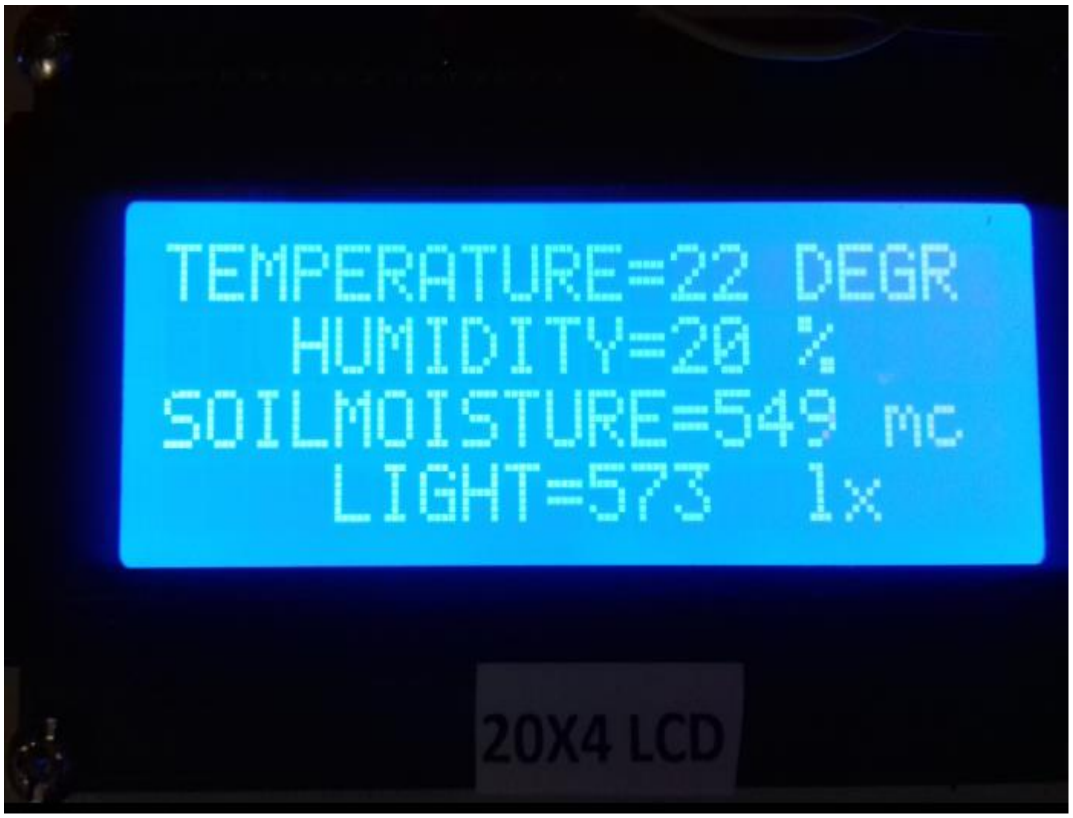


Figure 3.21: LCD displays information about data collected by sensor

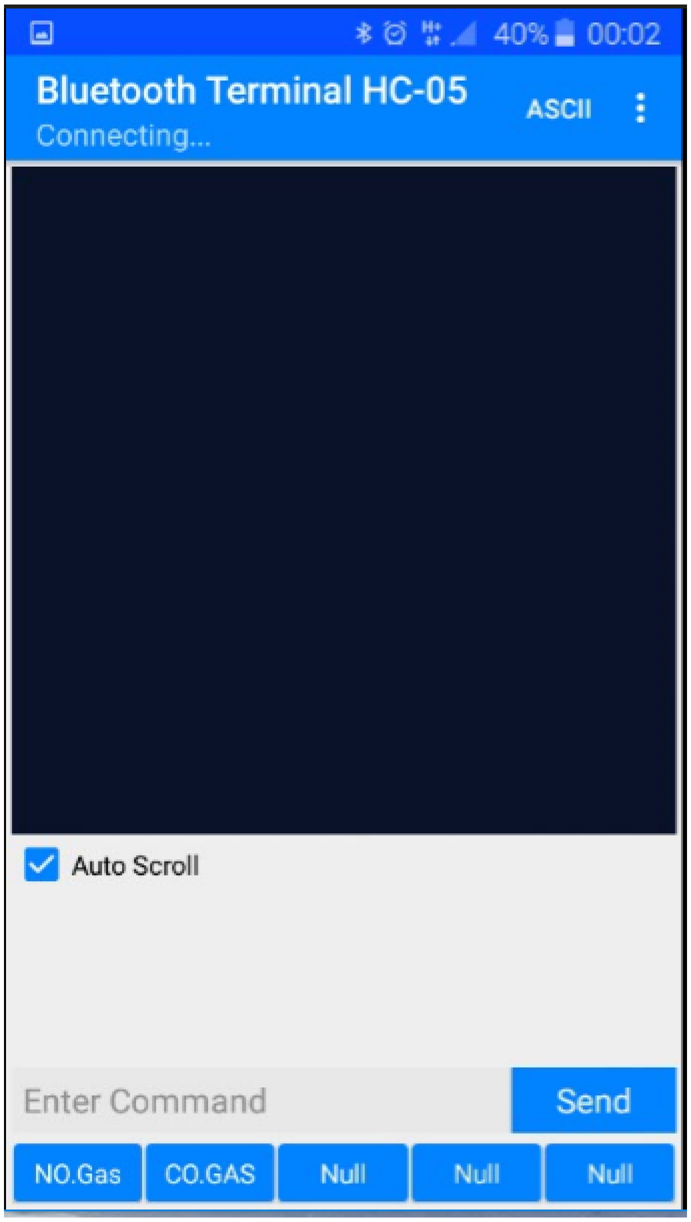


Figure 3.22: HC-05 Bluetooth terminal

Since designing an intelligent-easy to use system is one of the objectives of this study, an HC-05 Bluetooth module was used to receive commands from the Samsung J3 smart phone's Bluetooth terminal. The control commands of the Bluetooth terminal were '1' or '0'. When these commands were sent to the HC-05 Bluetooth module, the LCD changed from display shown in Figure 3.21 to display whether the carbon monoxide gas was detected or not (see Figure 3.23 and Figure 3.24). The Bluetooth terminal shown in Figure 3.22 is downloadable from Google play store.

When the “NO. Gas” tab was clicked in the Bluetooth terminal, command to change the LCD display was sent. Whenever CO gas was present, the LCD would not respond to the command and its screen would not change. When there was no CO gas detected, then the HC-05 would respond to the command and display information as shown Figure3.23. This display would be shown for five seconds before LCD returned to the display of sensor data as shown in Figure 3.21.

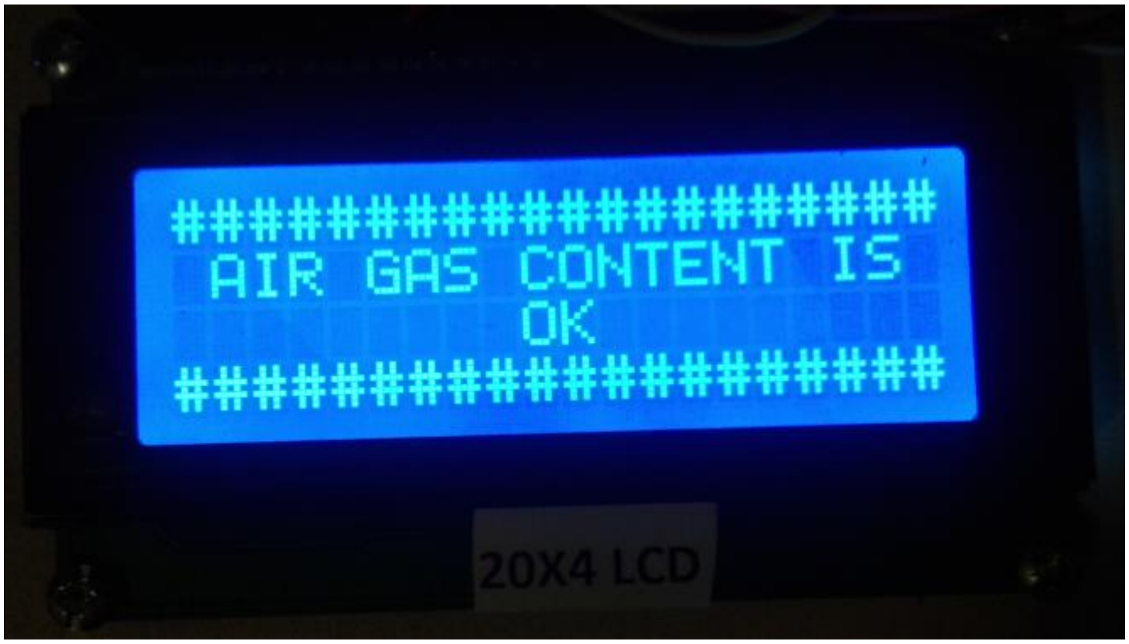


Figure 3.23: LCD display if there is no CO gas detected

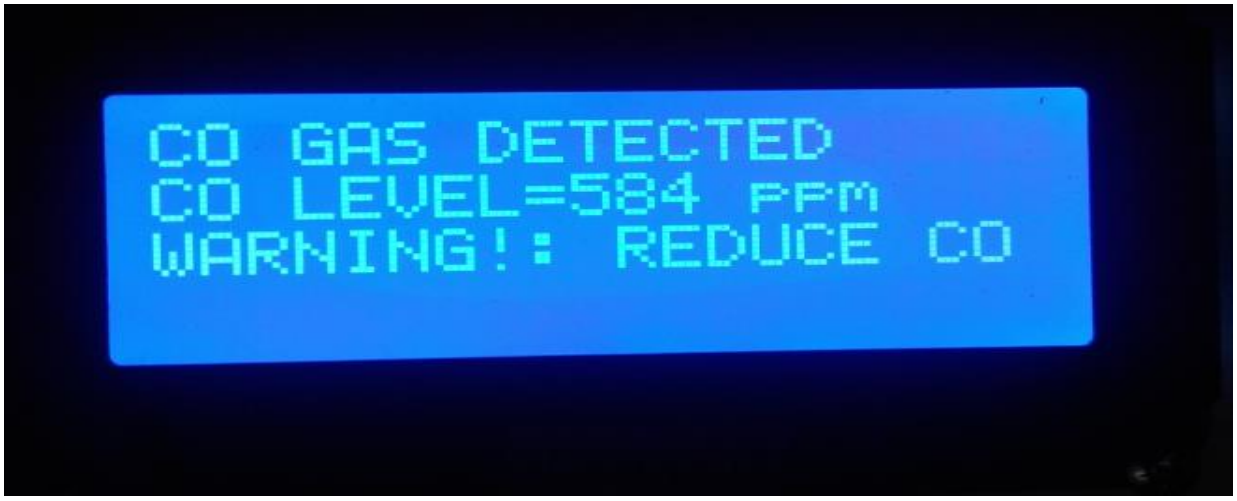


Figure 3.24: LCD display if CO gas is detected

When the “CO. Gas” tab was clicked in the Bluetooth terminal, command to change the LCD display would be sent to HC-5 Bluetooth module. Whenever CO gas was detected, the LCD would change to a display shown in Figure 3.24. This display would be shown for five seconds before LCD returned to the display of sensor data as shown in Figure 3.21. However, if there was no CO gas detected on click of “CO. Gas” tab, the HC-05 would not respond to the command and the LCD screen would not change from the display shown in Figure 3.21.

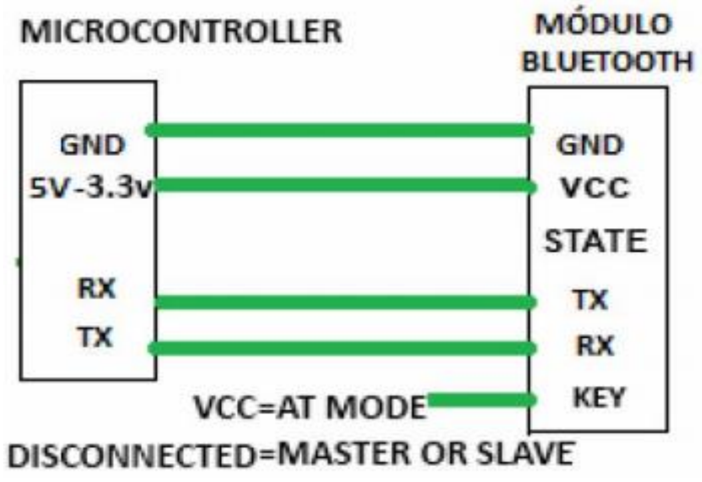


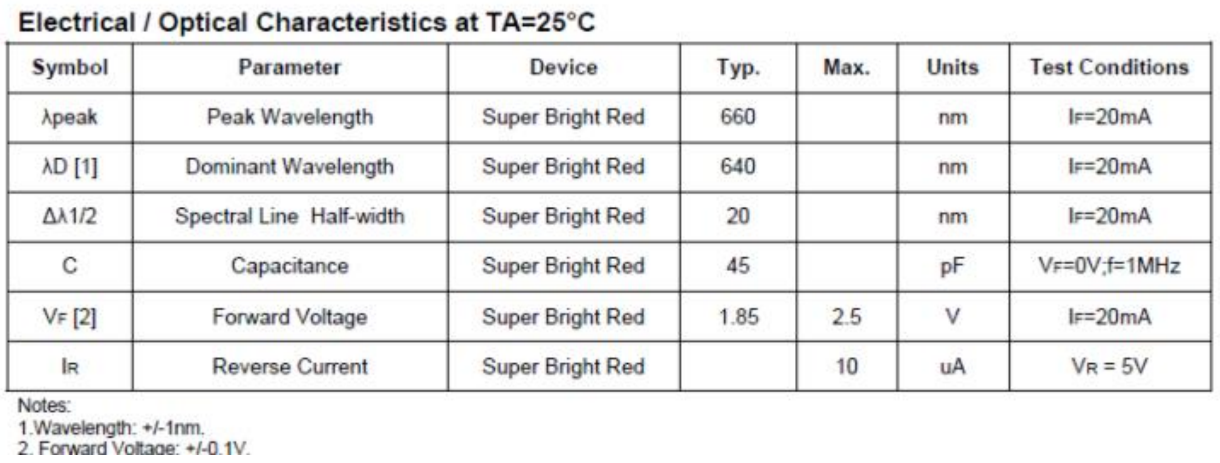
Figure 3.25: HC-05 Typical application circuit

Figure 3.25 shows the typical circuit for connecting HC-05 module. To get the module to AT mode the key stick must be associated with VCC.

3.3.2. Function of the Led in the coordinator station

The blinking LED connected to digital output pin 13 of the Arduino microcontroller in the coordinator station indicated that data was successfully received from the sensor station. If the blinking stopped then that meant sensor data was not received from the sensor station.

Table 3.5: Led specifications



Electrical / Optical Characteristics at TA=25°C

Symbol	Parameter	Device	Typ.	Max.	Units	Test Conditions
λpeak	Peak Wavelength	Super Bright Red	660		nm	I_F=20mA
λD [1]	Dominant Wavelength	Super Bright Red	640		nm	I_F=20mA
Δλ1/2	Spectral Line Half-width	Super Bright Red	20		nm	I_F=20mA
C	Capacitance	Super Bright Red	45		pF	V_F=0V;f=1MHz
V_F [2]	Forward Voltage	Super Bright Red	1.85	2.5	V	I_F=20mA
I_R	Reverse Current	Super Bright Red		10	uA	V_R = 5V

Notes:
1.Wavelength: +/-1nm.
2. Forward Voltage: +/-0.1V.

3.3.3. Function of the relay module in the coordinator station

The relay module has eight relays some of which were used for driving heater, humidifier, fan, sprinkler and light bulb. CO_2 gas was injected whenever the LCD instruction was to reduce CO gas as displayed by the LCD in Figure 3.24. The relay output that was active was indicated by red LED which corresponded to that particular relay. Figure 3.27 shows the red LED of the relay module illuminated to indicate that the second relay output from bottom view was the one activated. If more than one relay outputs were to be activated, then the corresponding number of LEDs would glow red.

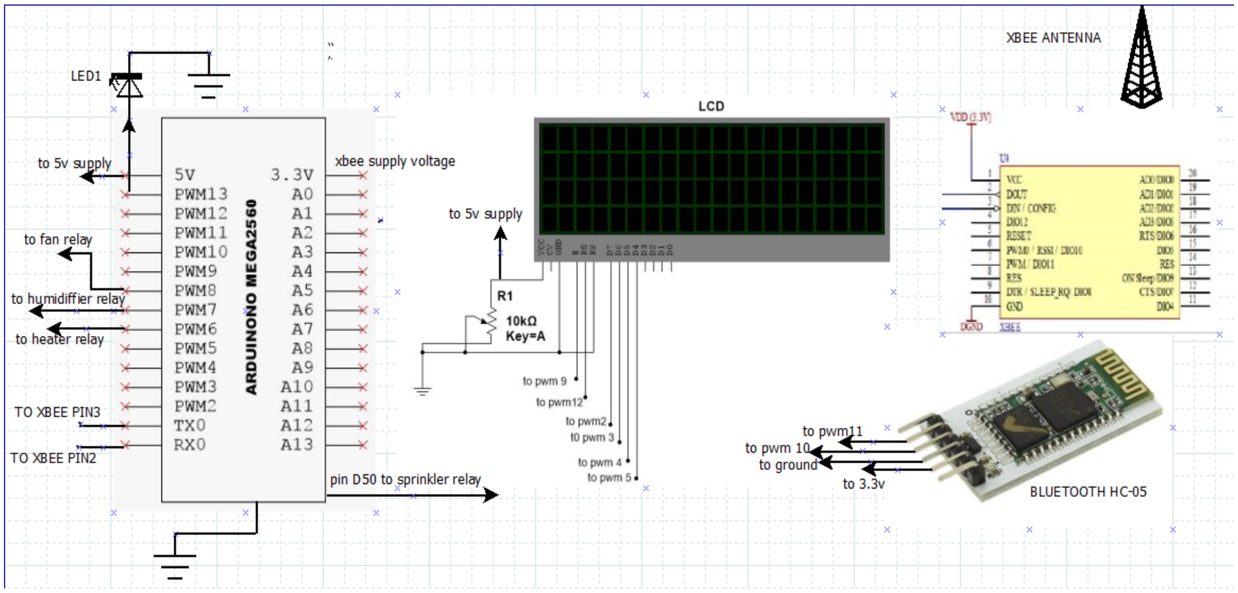


Figure 3.26: Circuit diagram of the coordinator station

Figure 3.27 shows a complete coordinator station that was designed for this research study.

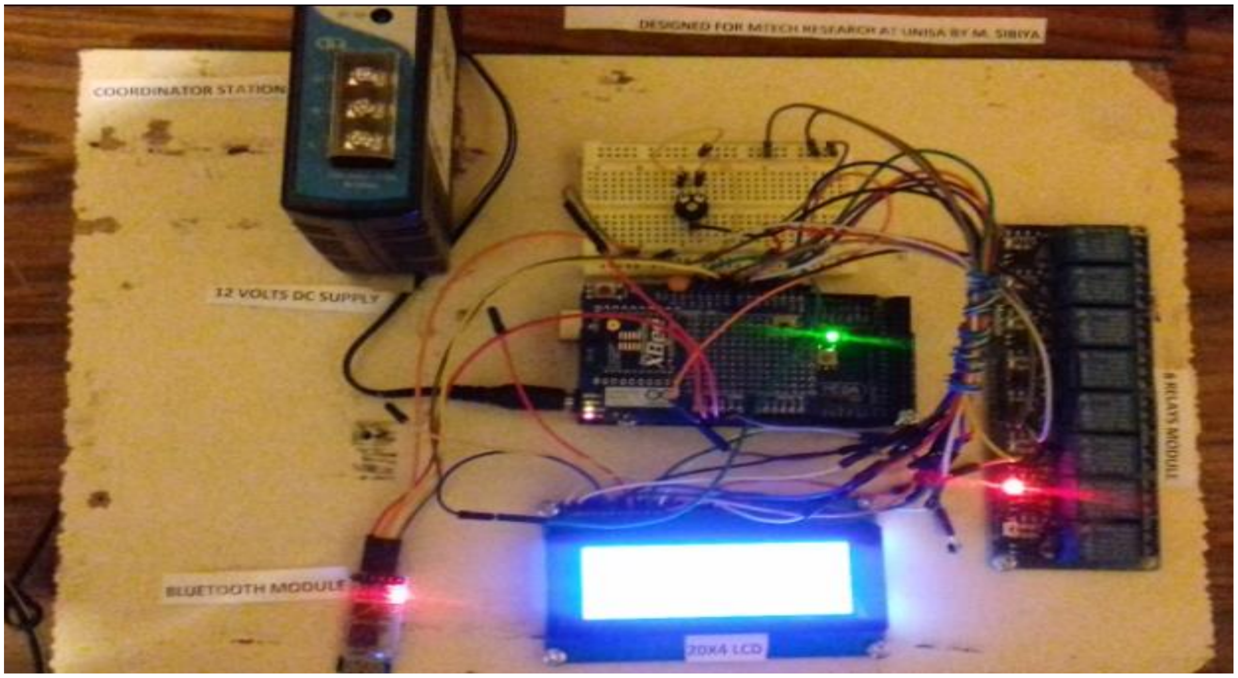


Figure 3.27: Designed coordinator station

3.4 Central station design

The circuit diagram for central station is shown in Figure 3.28. The Arduino code for the control algorithm is included in Appendix C. As it can be seen in Figure 3.29, the circuit was connected to the USB port of the personal computer so that sensor data plots could be viewed from special program that was designed in Python programming language. Python programming language is a high level language that has powerful libraries that are downloadable from Python org site [30]. The Python code with important libraries is included in Appendix D.

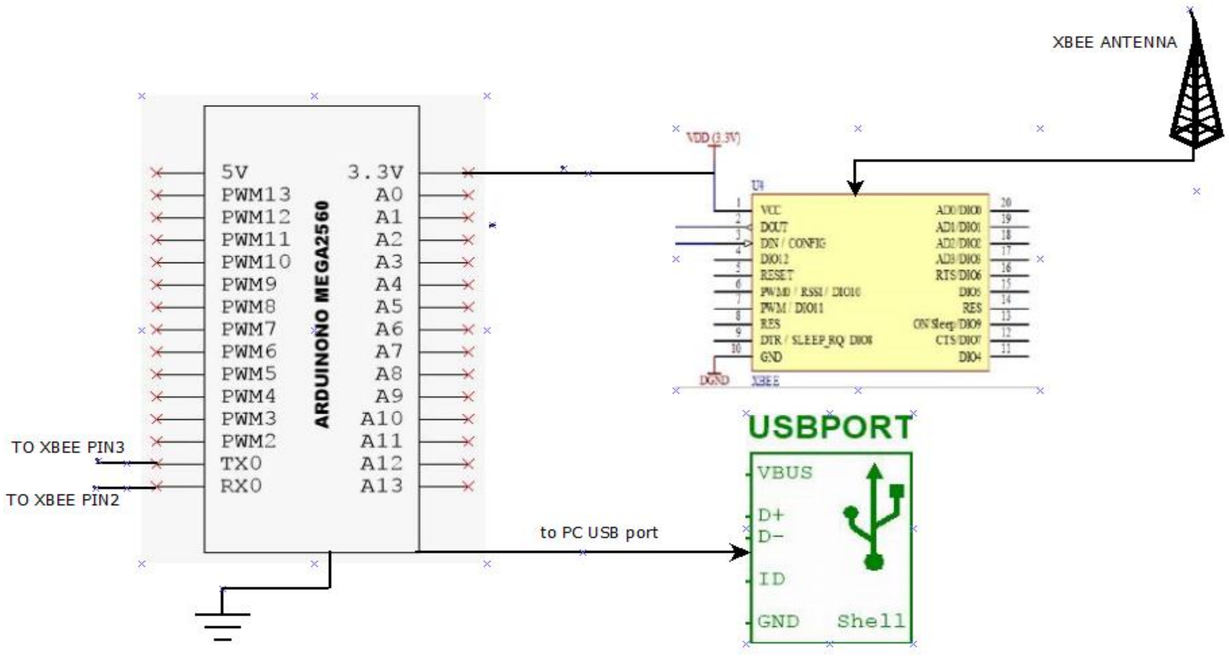


Figure 3.28: Circuit diagram of the central station

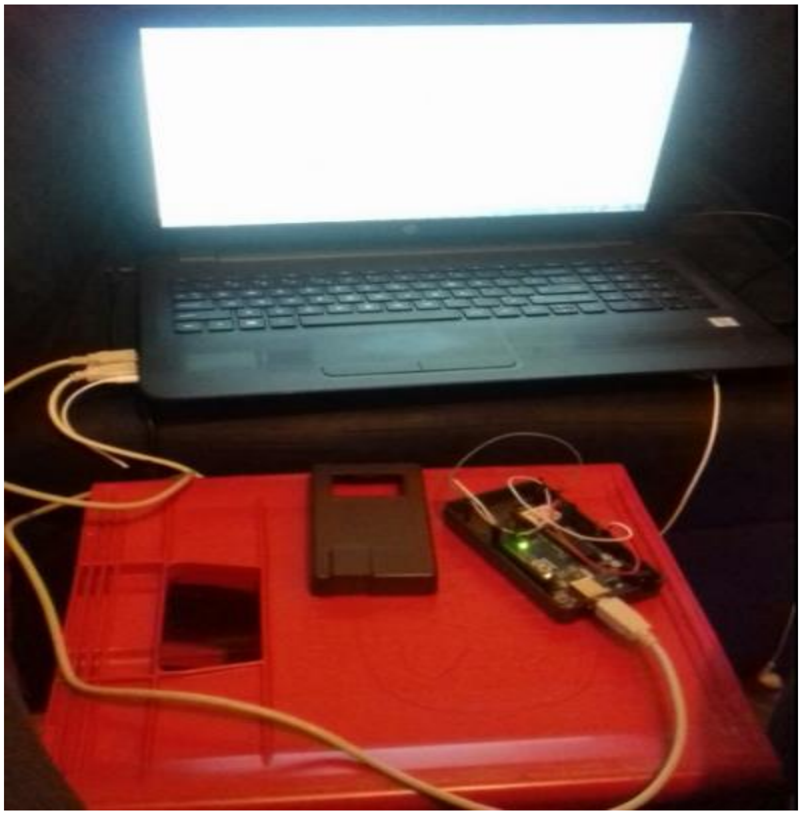


Figure 3.29: Designed central station

3.5 Arduino libraries

Arduino IDE is an open source environment for writing Arduino programs known as sketches. For this reason, the Arduino microcontroller gained popularity and has been used by hobbyists and Scientists in many research areas. This led to a need for libraries to be developed to speed up the design processes. Below is a list of Arduino libraries that were used in this research study.

i. DHT11 sensor library [31]
ii. Arduino Fuzzy logic library [32]

After downloading these libraries from the internet, they were then installed in the Arduino IDE and ready for use.

3.6 Python libraries

Python is an open source high level programming language. For its variety of libraries and simplicity, Python has become high level programming language that many fields use for their applications. In this research study, Python was used to write a program that generated plots of live streaming sensor data. Important libraries needed for development of this program that were downloaded from Python organization [30] are as listed below.

i. Numpy library
ii. Drawnow library
iii. Serialpy library
iv. Matplotlib library

To install these libraries using the windows command terminal, PIP which is a program for installing Python libraries was first installed in Python directory. In order to view sensor data plots, the user had to connect the central station to the USB port of the personal computer and then opened Python code with "VDLE for Python (Python integrated development environment)".

3.7 Experiment 1: Measuring soil moisture content using grove moisture sensor

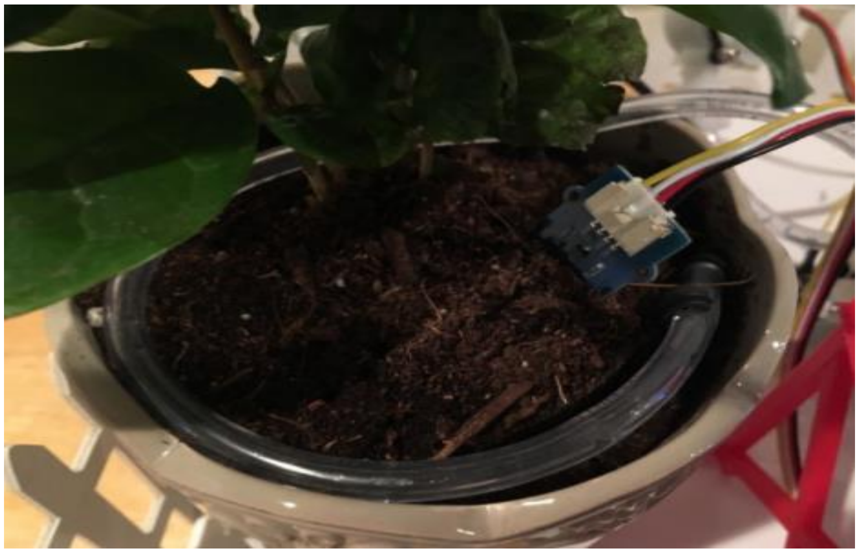


Figure 3.30: Grove moisture sensor inserted in soil to measure soil moisture content

By inserting the grove moisture sensor in different pot plants of different soil moistures gave results that varied depending on the degree of the soil moisture (dryness or dampness of the soil, see Figure 3.30). Dry soil results were first read when the moisture sensor was inserted in dry soil. After getting results from the dry soil water was then poured into the dry soil. With grove moisture sensor still inserted in the soil, graph results changed and gave expected results in line with the conditions of the wet soil. The moisture content was measured in analog readings of the microcontroller steps.

3.8 Experiment 2: Measuring temperature and humidity using DHT11 sensor

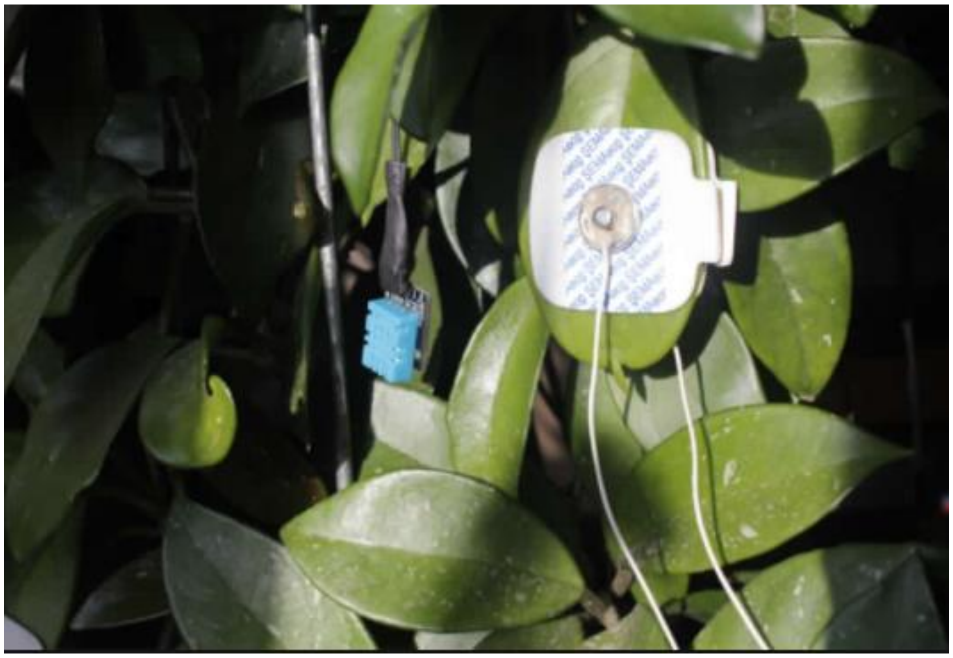


Figure 3.31: DHT11 sensor measuring temperature and humidity in a greenhouse

To change temperature and humidity of the surrounding environment the hair dryer was used. A hair dryer could be set to different degrees of temperature using a rotary button. The speed of the fan could also be set in order to change humidity to different values. A hair dryer was hovered over the DHT11 in order to vary temperature and humidity (see Figure 3.31).

As temperature values were changed, humidity values changed accordingly as temperature and humidity are relative to each other. The temperature was measured in degrees Celsius and humidity was measured in percentage of concentration.

3.9 Experiment 3: Measuring Carbon monoxide content using MQ7 gas sensor

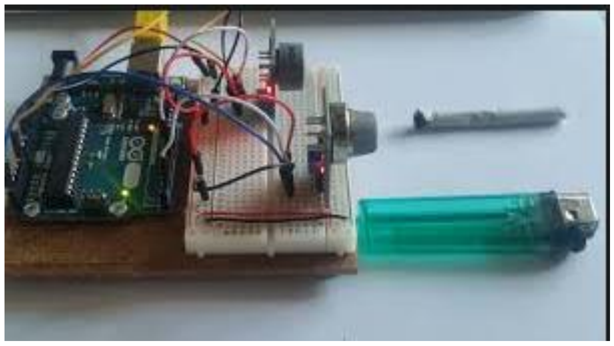


Figure 3.32: MQ7 sensor used to measure carbon monoxide

MQ7 sensor was used to measure levels of carbon monoxide in the green house. To conduct this experiment, cigarette lighter gas was released around the MQ7 gas sensor (see Figure 3.32). The cigarette lighter has higher concentration of carbon monoxide which was the reason why the cigarette lighter was chosen as the source of carbon monoxide gas. The longer the cigarette lighter was held closer to MQ7 gas sensor, the higher the concertation of carbon monoxide. The measured carbon monoxide was given in analog reading which was then converted to ppm by the use of the appropriate formula.

3.10 Experiment 4: Measuring light intensity using LDR

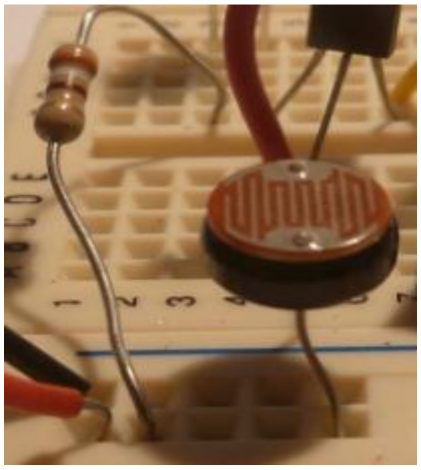


Figure 3.33: Light sensor(LDR) used to measure light intensity

To conduct this experiment, three different lids that differed in opaqueness were used. The three different lids were used to cover up the light sensor (see Figure 3.33). Since the lids were used independently, the result was a graphic plot that changed accordingly depending on the opaqueness of the lid used. The measured results were given in analog reading of the microcontroller steps.

3.11 Experiment 5: Demonstrate fuzzy logic control of temperature, humidity, moisture content, Carbon monoxide concentration and light intensity

The generated plots of sensor data could not demonstrate fuzzy logic control abilities of the system. The experiments to demonstrate fuzzy logic control could only be conducted by either using MATLAB or LabVIEW programs. MATLAB results were straight forward and easily interpretable. For this reason, MATLAB was chosen for fuzzy logic analysis of results.

3.12 Chapter summary

In this chapter we explained, in detail, the building components of the project and their specifications. The schematic diagrams for sensor station, coordinator station and central station were scrutinized and the operation of each was clearly explained. Necessary Python libraries for building data display program in the central station were also explained. Finally, the conduction of experiments and testing procedures were also explained.

CHAPTER 4: Presentation of experiments results and their analysis

4.1 Introduction

The specific objectives of this study were:

i. To determine, design and develop a low-cost microcontroller for monitoring of climatic parameters in a greenhouse using smart technology.
ii. To use fuzzy logic to optimize the control algorithms of a system monitoring a greenhouse.
iii. To specify the necessary climatic parameters for humidity, temperature, soil moisture and sunlight for sensors.

This research was conducted based on research questions which so far have led to the success of the system’s experiments. These research questions contributed a lot on the findings that are presented in this chapter and were:

i. Which micro-controller board is going to be used in the project?
ii. Which wireless radio and protocol are going to be used?
iii. How is fuzzy logic inference going to be simulated?
iv. How is data going to be viewed?
v. How the research system results are compared to basic systems?

4.2 Experiment 1 results: Analysis of moisture sensor data plots

Each microcontroller pin has maximum analog reading of 1023 steps. With appropriate formulas, the user could use this information to calculate the desired parameters of voltage.

Voltage = (Analog reading/1023) *5 is the formula that could be used to calculate voltage at a known value of analog reading. The values for humidity and temperature were calculated in the DHT11 library of the Arduino. In that case, the values of the temperature and humidity were read in their true form with temperature measured in degrees Celsius and humidity in percentage values. The moisture sensor used in this study has specifications as tabulated in Table 4.1.

Table 4.1: Specifications of grove moisture sensor

Analog reading value in dry soil	***Analog reading value in damp soil***	***Analog reading in wet soil***
0-300	300-700	700-950

Table 4.2: Moisture sensor analog readings and corresponding voltages as read from the moisture sensor data plots.

Moisture sensor data plot values(Analog readings)	***Calculated output voltage values***	***Type of soil detected***	***Action taken by coordinator station***
0	0 volts	Dry soil	Sprinklers are on
710	3.47 volts	Wet soil	Sprinklers are off
352	1.7 volts	Damp soil	Sprinklers are off

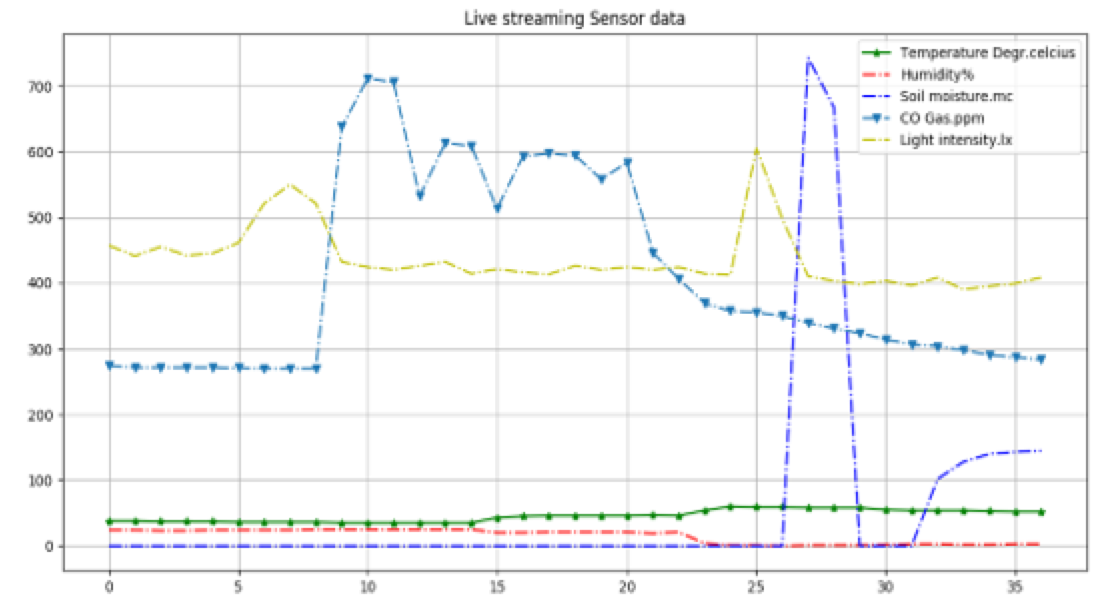


Figure 4. 1: Moisture sensor plots

Soil moisture plots are shown in Figure 4.1 (see keys). Dry soil results are shown between sample values 0 to 26 (See Figure 4.1). It can be seen that the analog reading was 0 between these sample values. The graph changes between sample values of 26 and 27, and gets to a peak analog value of 710. A peak analog value of 710 was recorded when the soil was wet. Between sample values of 26 to 29 the graph's analog value decreases to analog values of damp soil until the dry soil values are reached. The changing soil moisture conditions activated or deactivated the relay that was driving the sprinkler. When the soil moisture sensor was inserted in dry soil, the sprinkler relay was activated and deactivated when the soil moisture sensor was inserted in humid to wet soil. The decision making was based on fuzzy logic rules (see Experiment 5).

4.3 Experiment 2 results: Analysis of humidity and temperature data plots

Analysis of temperature sensor data plots

Table 4.3: Temperatures in degrees Celsius as read from temperature sensor data plots of Figure 4.2

Temperature sensor data plot values(degrees Celsius)	***Temperature classification for fuzzy logic***	***Action taken by coordinator station***
60℃	hot	See fuzzy rules in experiment 5
26℃	normal	See fuzzy rules in experiment 5
17℃	cold	See fuzzy rules in experiment 5

Fuzzy rules explained in experiment 5 show the decisions that were made by the coordinator station. The changing temperature conditions controlled the heater and the fan according to the fuzzy logic rules that were designed for the system (see Experiment 5).

Analysis of humidity sensor data plots

Table 4.4: Humidity percentages as read from humidity sensor data plots of Figure 4.2

Humidity sensor data plot values(% of humidity)	***Humidity classification for fuzzy logic***	***Action taken by coordinator station***
85%	high	See fuzzy rules in experiment 5
77%	optimal	See fuzzy rules in experiment 5
17%	low	See fuzzy rules in experiment 5

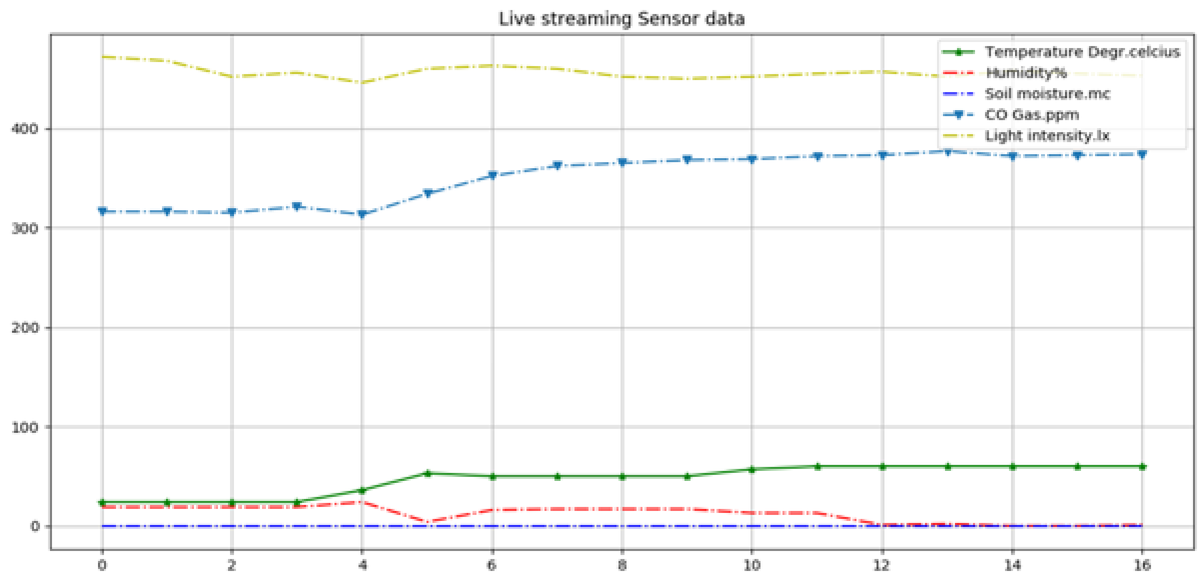


Figure 4. 2: Temperature and Humidity plots

Figure 4.2 shows temperature and humidity plots (see keys). It can be seen in Figure 4.2 that as the temperature was increased, humidity tended to decrease. This relationship between temperature and humidity is so obvious between sample values of 4 and 6 (see Figure 4.2). Between these sample values the maximum temperature read was 60℃ while humidity approached 5%. The changing humidity conditions controlled the humidifier according to the fuzzy logic rules that were designed for the system (see Experiment 5).

4.4 Experiment 3 results: Analysis of carbon monoxide data plots

Table 4.5: Carbon monoxide analog readings and corresponding voltages as read from gas sensor data plots of Figure 4.3

Carbon monoxide sensor data plot values(Analog readings)	***Calculated voltage values***	***Status of gas detected***	***Action taken by the user***
0	0 volts	CO gas not detected	CO gas levels are fine
290	1.417 volts	CO gas not detected	CO gas levels are fine
510	2.492 volts	CO gas detected	Inject carbon dioxide to neutralize carbon monoxide concentration
710	3.47 volts	CO gas detected	Inject carbon dioxide to neutralize carbon monoxide concentration

If the CO gas's analog value was above 500, carbon dioxide (CO_2) enough to neutralize Carbon monoxide(CO) gas was injected.

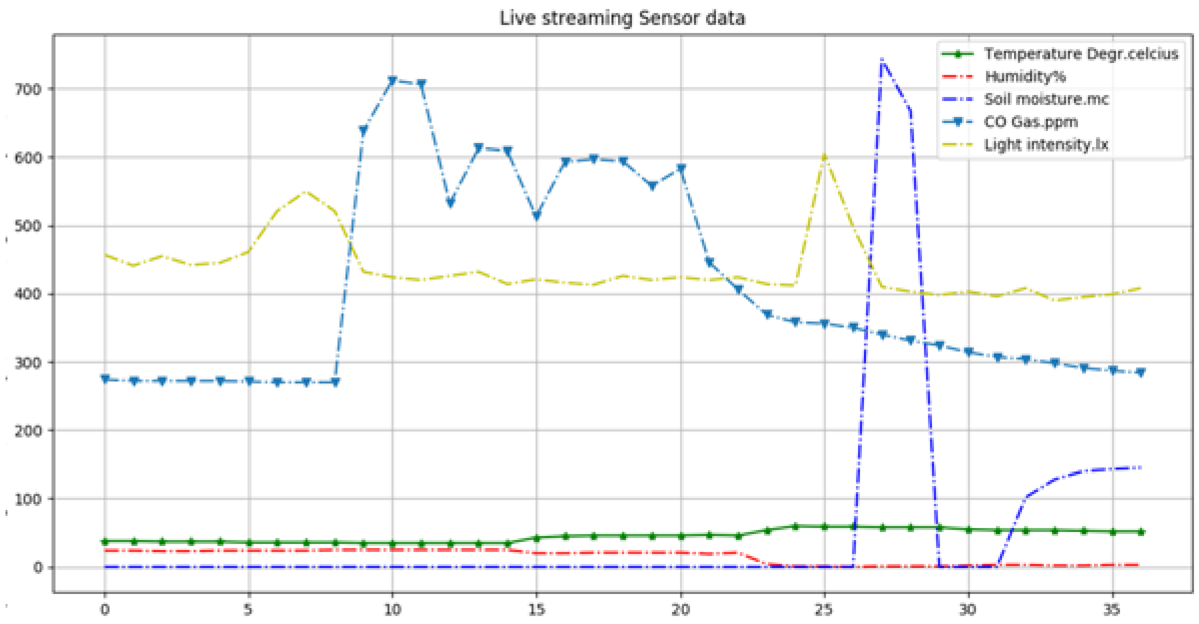


Figure 4. 3: Carbon monoxide data plots

Figure 4.3 shows carbon monoxide data plots (see keys). Carbon monoxide was not a threat between the sample values of between 0 and 8 as it read around 280ppm.The levels of carbon monoxide were considered to be harmful to plant roots beyond a minimum of 500ppm. These harmful readings were recorded between sample values of 9 and 20 (see Figure 4.3). The changing carbon monoxide conditions controlled the CO_2 injector according to the fuzzy logic rules that were designed for the system.

4.5 Experiment 4 results: Analysis of light sensor(LDR) data plots

Table 4.6: Light intensity analog readings and corresponding voltages as read from light sensor data plots of Figure 4.4

Light sensor data plot values(Analog readings)	***Calculated voltage values***	***Physical condition detected***	***Action taken by coordinator station***
0	0 volts	Light detected	Bulb is off
450	2.199 volts	Light detected	Bulb is off
600	2.93 volts	Light detected	Bulb is off
650	3.176 volts	Darkness detected	Bulb is on
850	4.154 volts	Darkness detected	Bulb is on

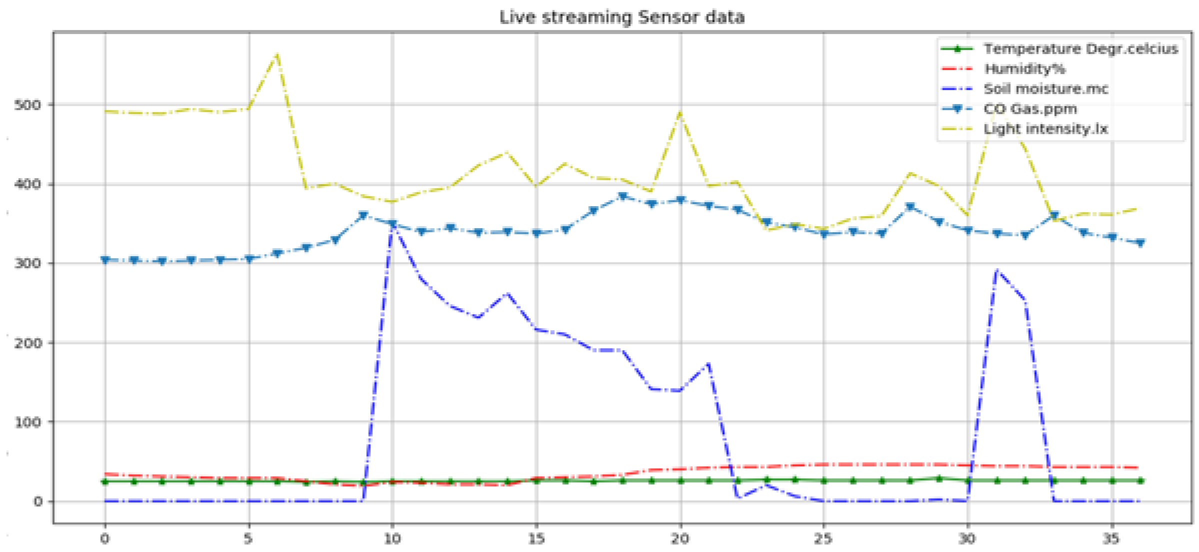


Figure 4. 4: Light intensity data plots

Figure 4.4 shows the light intensity data plots (see keys). The light intensity plot varies depending on the opaqueness of the lid that was used to cover up the light sensor. The darker lid that was used to cover up the light sensor resulted in higher analog readings which activated the bulb. Analog readings beyond an analog value of 500 were regarded as levels that sensed darkness. The changing light intensity conditions controlled the light bulb according to the fuzzy logic rules that were designed for the system (see Experiment 5).

4.6 Experiment 5: Fuzzy logic experiment results

4.6.1 Fuzzy logic results of temperature and humidity

In fuzzy logic the input variable is further classified into what is known as membership functions. Membership function can be seen as part of the input variable with a certain range limiting it within the chosen values of the available input variable. Figure 4.5 shows the available input variable of temperature in degrees written on the x-axis with maximum range of 100 °C. The blue triangle in Figure 4.5 covers temperatures from 0 to 25 °C which is a portion of input temperatures representing cold status. These temperatures were classified as "cold" membership function of the temperature input variable. The orange triangle in Figure 4.5 represents a "normal" membership function of the temperature input variable with temperatures ranging from 20°C to 32.5°C while the red trapezoid represents a "hot" membership function of temperatures from 27 °C to 100°C. The degree of membership is represented on the y axis of Figure 4.5. In "cold" membership function for example, the degree of membership (y axis value) was higher when temperature was 17°C than when it was 10°C.The knowledge of the degree of membership in a membership function was not important in this study as the system designed was not a system where accuracy was a matter of concern as it would be design of medical systems for instance.

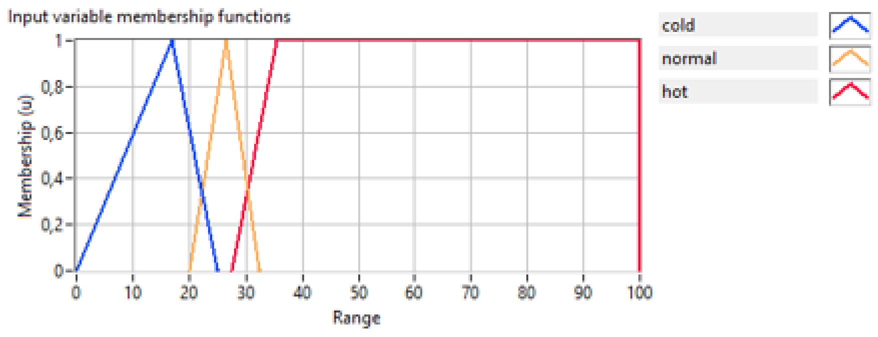


Figure 4. 5: Temperature input variable with cold, normal and hot membership functions

Rules were designed to control output variables when input temperatures fell under any of the membership functions. For example, we could set a rule that instructed the system to switch on the fan if the surrounding temperature was of a value belonging to "hot" membership function. This therefore would mean that, any input value of temperature within the trapezoid area could perform the latter rule. Since the change in temperature affected humidity, both the temperature and humidity inputs were considered simultaneously when control rules of the system were designed. Figure 4.6 shows the second input variable of the system which is humidity. This humidity had its values expressed as percentage on the x-axis. The maximum value of humidity the system could detect was 100%. This meant that the range on the x axis that was expressed in percentage (%) had a maximum humidity value of 100%. As previously mentioned, the y axis represents the degree of membership (How much of the input value was associated with membership function of the input variable). The humidity value detected by the system's sensors was realized as a value belonging to a particular membership function of the humidity input variable. Any value of humidity within the range of 0 to 75% undoubtedly belonged to "low" humidity membership function. This group of values that represented low humidity was known as "low" humidity membership function. 75% to 85% humidity range represented "optimal" humidity membership function while 85% to 100% range represented "high" humidity membership function. In this study the whole concept of how these inputs were viewed in fuzzy logic sense is shown below:

Temperature input variable = ("cold "membership function + "normal" membership function + "hot" membership function). In binary logic (also known as traditional logic) the input could be classified only into two states of either high or low. This was different in fuzzy logic because the input could be classified into many states depending on the system designer's choice. A simple example is that of a person when his height is to be judged. His height could be classified only as either tall or short in binary logic but in fuzzy logic the same person's height could be classified as short, medium short, tall, medium tall and so forth.

Humidity input variable = ("low "membership function + "optimal" membership function + "high" membership function)

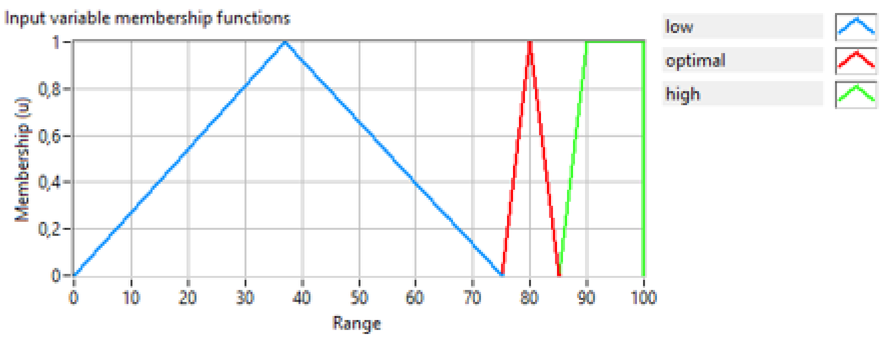


Figure 4. 6: Humidity input variable with low, optimal and high membership functions

To summarize, this research study system had temperature input and humidity input that needed to be considered simultaneously when the control rules were designed. The temperature input was fuzzified (classified) into three membership functions of "cold" temperature, "normal" temperature and "hot" temperature. The humidity input variable was fuzzified (classified) into "low" humidity, "optimal" humidity and "high" humidity. The output variable was needed to set the reference point when the rules are designed. Figure 4.7 shows the membership functions of the output variable as "phase1" to "phase9". When rules were designed, the system looked at the input values through its inputs and made decisions based on the design rules. The output result had to be any of the output membership functions from phase1 to phase 9. The output range in this study is represented in samples of unit steps on the x-axis (These are just steps that are intended to make reference scaling for successful design of fuzzy inference control system). The output variable was fuzzified (classified) into 9 phases which were output destinations of the control rules.

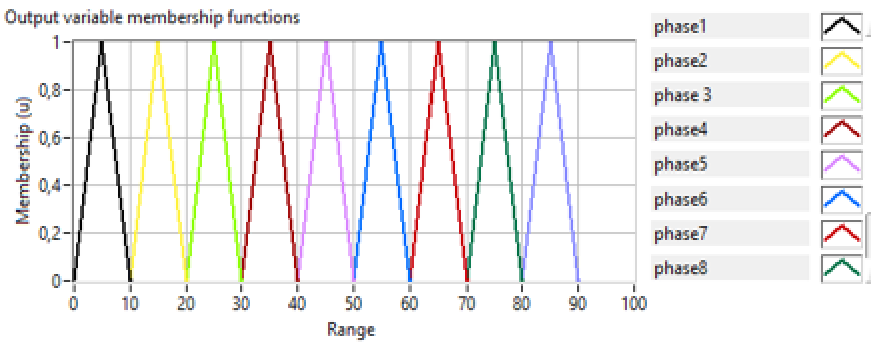


Figure 4. 7: Output variable called "reference output" with phase1-phase9 membership functions

The designed control rules of temperature and humidity are shown below:

1. If TEMPERATURE is HOT and HUMIDITY is HIGH, then OUTPUT is PHASE1
2. If TEMPERATURE is NORMAL and HUMIDITY is HIGH, then OUTPUT is PHASE2
3. If TEMPERATURE is COLD and HUMIDITY is HIGH, then OUTPUT is PHASE3
4. If TEMPERATURE is HOT and HUMIDITY is OPTIMAL, then OUTPUT is PHASE4
5. If TEMPERATURE is NORMAL and HUMIDITY is OPTIMAL, then OUTPUT is PHASE5
6. If TEMPERATURE is COLD and HUMIDITY is OPTIMAL, then OUTPUT is PHASE6
7. If TEMPERATURE is HOT and HUMIDITY is LOW, then OUTPUT is PHASE7
8. If TEMPERATURE is NORMAL and HUMIDITY is LOW, then OUTPUT is PHASE8
9. If TEMPERATURE is COLD and HUMIDITY is LOW, then OUTPUT is PHASE9

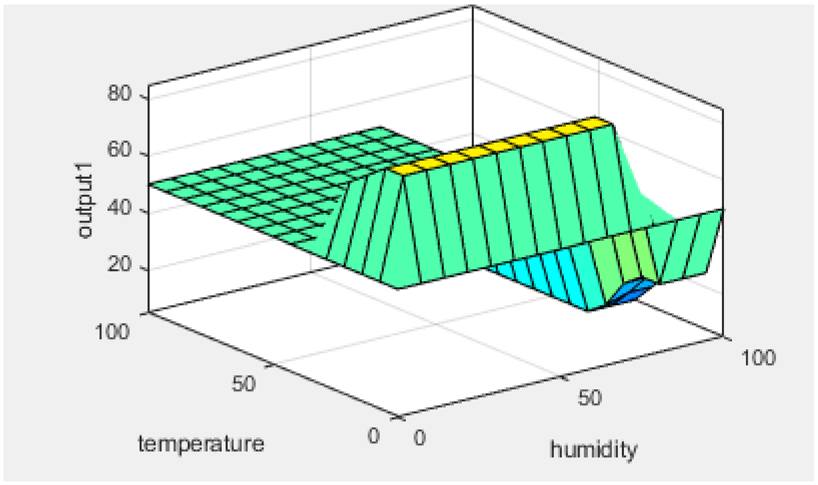


Figure 4. 8: Surface view Z-output of temperature input y-axis and humidity input x-axis

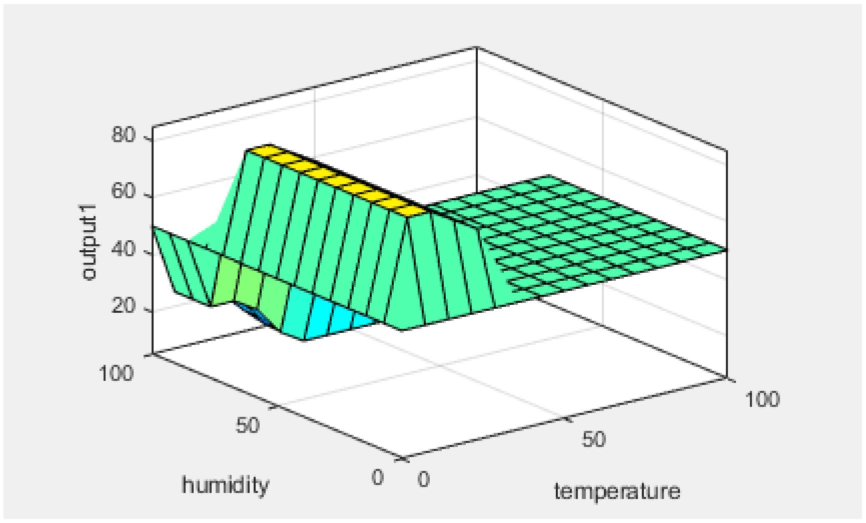


Figure 4. 9: Surface view Z-output of humidity input y-axis and temperature input x-axis

For rule1 to be invoked, the input temperature had to be of any value in the range of 27.5℃ to 100℃ (This was “hot” membership function for temperature input) while the input humidity had to be of any value in the range of 85% to 100% (This was “high” membership function for humidity input). Figure 4.10 shows the output results at these inputs. It can be seen that the output of 5 unit steps (PHASE 1 value of reference output) resulted when the temperature was 47.2℃ (“hot” membership function value of temperature input) and humidity was 92.3% (“high” membership function value of the humidity input). These outcomes satisfied design rule 1.

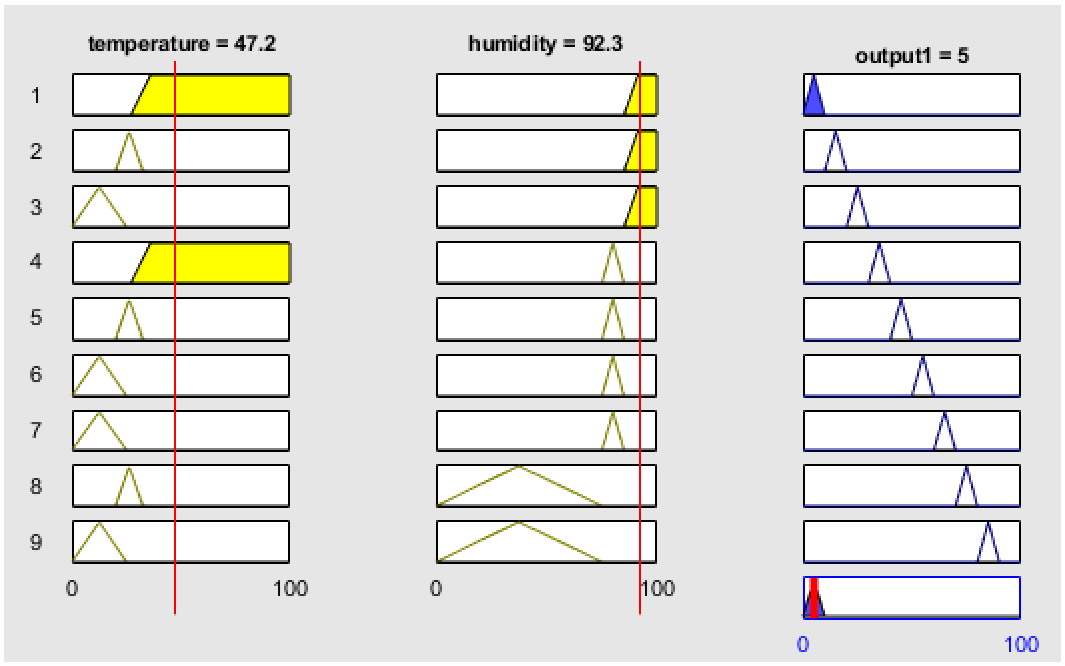


Figure 4. 10: Output results when rule 1 was invoked (temperature and humidity fuzzy logic analysis)

For rule 2 to be invoked, the input temperature had to be of any value in the range of 20℃ to 32.5℃ (This was "normal" membership function for temperature input) while the input humidity had to be of any value in the range of 85% to 100% (This was "high" membership function for humidity input). Figure 4.11 shows the output results of the inputs. It can be seen that the output of 15 unit steps (PHASE 2 value of reference output) resulted when the temperature was 25.2℃ ("normal" membership function value of temperature input) and humidity was 92.3% ("high" membership function value of the humidity input). These outcomes satisfied design rule 2.

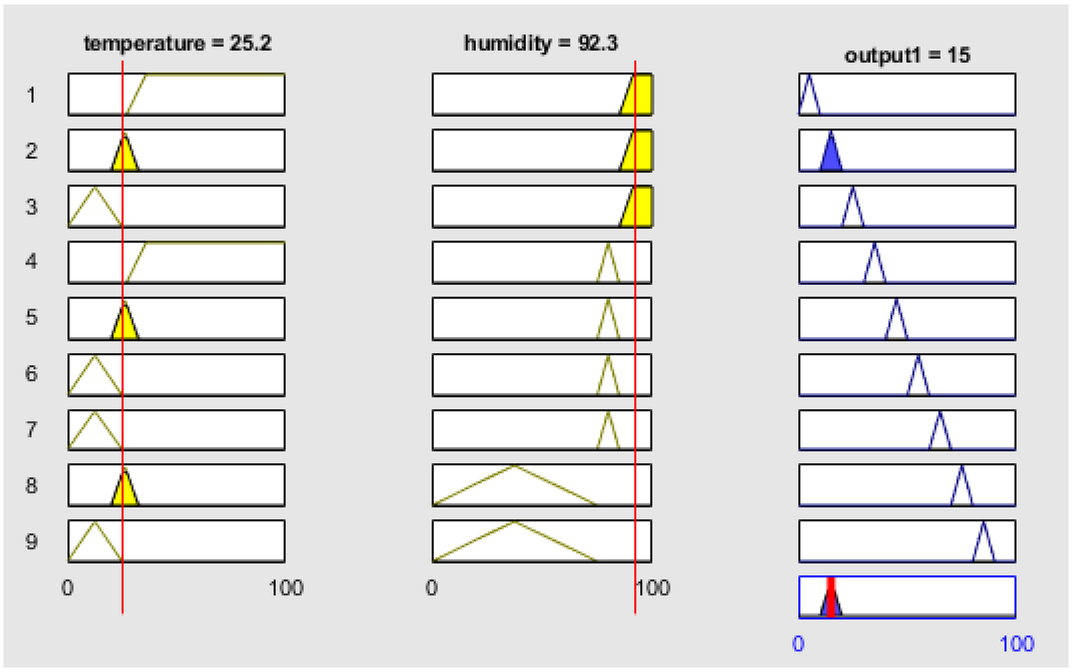


Figure 4. 11: Output results when rule 2 was invoked (temperature and humidity fuzzy logic analysis)

For rule 3 to be invoked, the input temperature had to be of any value in the range of 0℃ to 25℃ (This was "cold" membership function for temperature input) while the input humidity had to be of any value in the range of 85% to 100% (This was "high" membership function for humidity input). Figure 4.12 shows the output results at these inputs. It can be seen that the output of 25 unit steps (PHASE 3 value of reference output) resulted when the temperature was 5.96℃ ("cold" membership function value of temperature input) and humidity was 94.1% ("high" membership function value of the humidity input). These outcomes satisfied design rule 3.

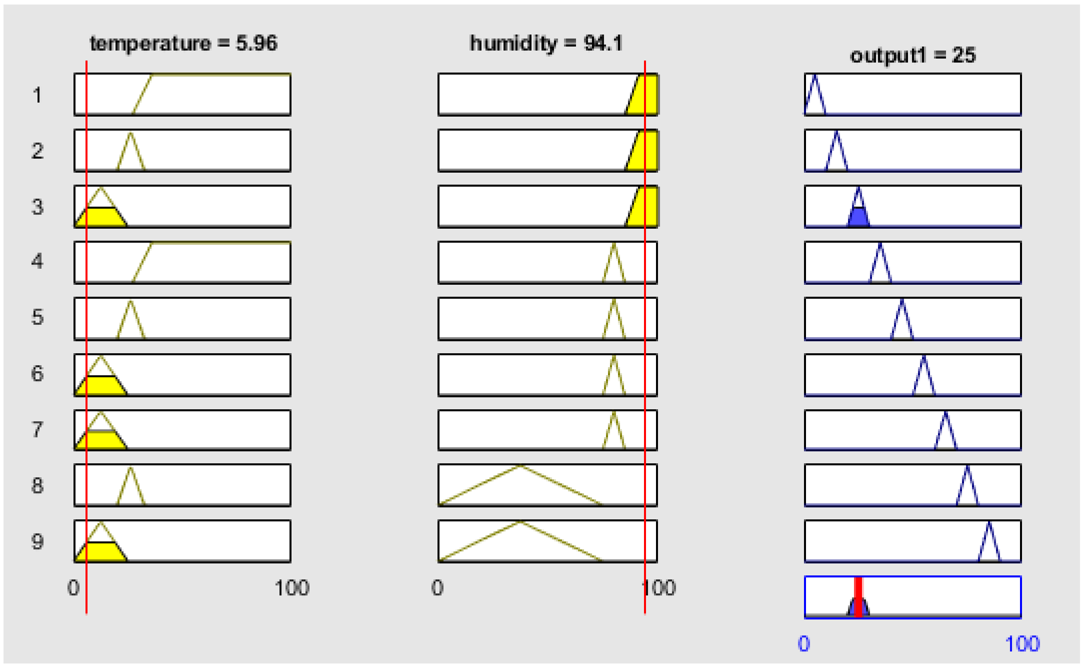


Figure 4. 12: Output results when rule 3 was invoked (temperature and humidity fuzzy logic analysis)

For rule 4 to be invoked, the input temperature had to be of any value in the range of 27.5℃ to 100℃ (This was “hot” membership function for temperature input) while the input humidity had to be of any value in the range of 75% to 85% (This was “optimal” membership function for humidity input). Figure 4.13 shows the output results at these inputs. It can be seen that the output of 37.4 unit steps (PHASE 4 value of reference output) resulted when the temperature was 31.7℃ (“hot” membership function value of temperature input) and humidity was 80.5% (“optimal” membership function value of the humidity input). These outcomes satisfied design rule 4.

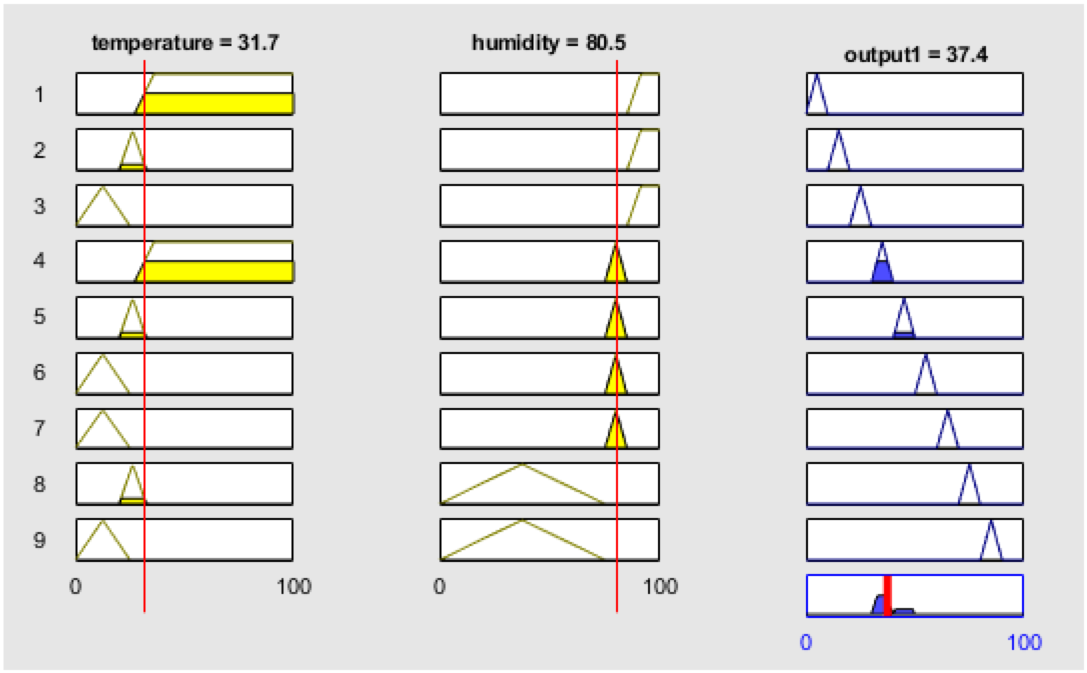


Figure 4. 13: Output results when rule 4 was invoked (temperature and humidity fuzzy logic analysis)

For rule 5 to be invoked, the input temperature had to be of any value in the range of 20℃ to 32.5℃ (This is "normal" membership function for temperature input) while the input humidity had to be of any value in the range of 75% to 85% (This was "optimal" membership function for humidity input). Figure 4.14 shows the output results at these inputs. It can be seen that the output of 48 unit steps (PHASE 5 value of reference output) resulted when the temperature was 24.3℃ ("normal" membership function value of temperature input) and humidity was 82.3% ("optimal" membership function value of the humidity input). These outcomes satisfied design rule 5.

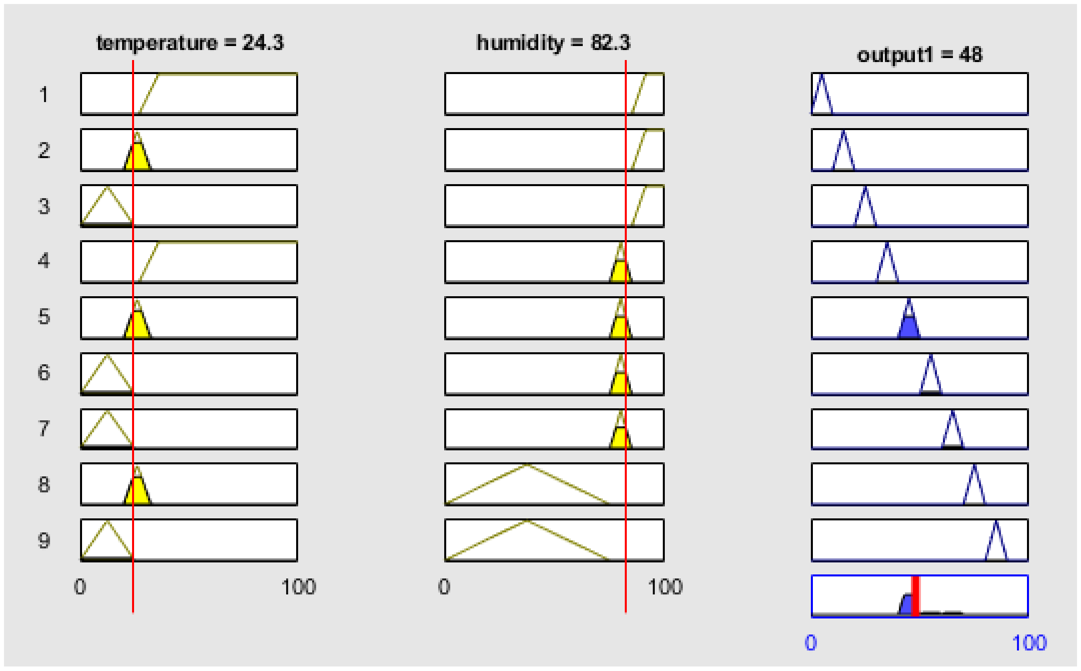


Figure 4. 14: Output results when rule 5 was invoked (temperature and humidity fuzzy logic analysis)

For rule 6 to be invoked, the input temperature had to be of any value in the range of 0℃ to 25℃ (This is “cold” membership function for temperature input) while the input humidity had to be of any value in the range of 75% to 85% (This was “optimal” membership function for humidity input). Figure 4.15 shows the output results at these inputs. It can be seen that the output of 60 unit steps (PHASE 6 value of reference output) resulted when the temperature was 7.8℃ (“cold” membership function value of temperature input) and humidity was 79.5% (“optimal” membership function value of the humidity input). These outcomes satisfied design rule 6.

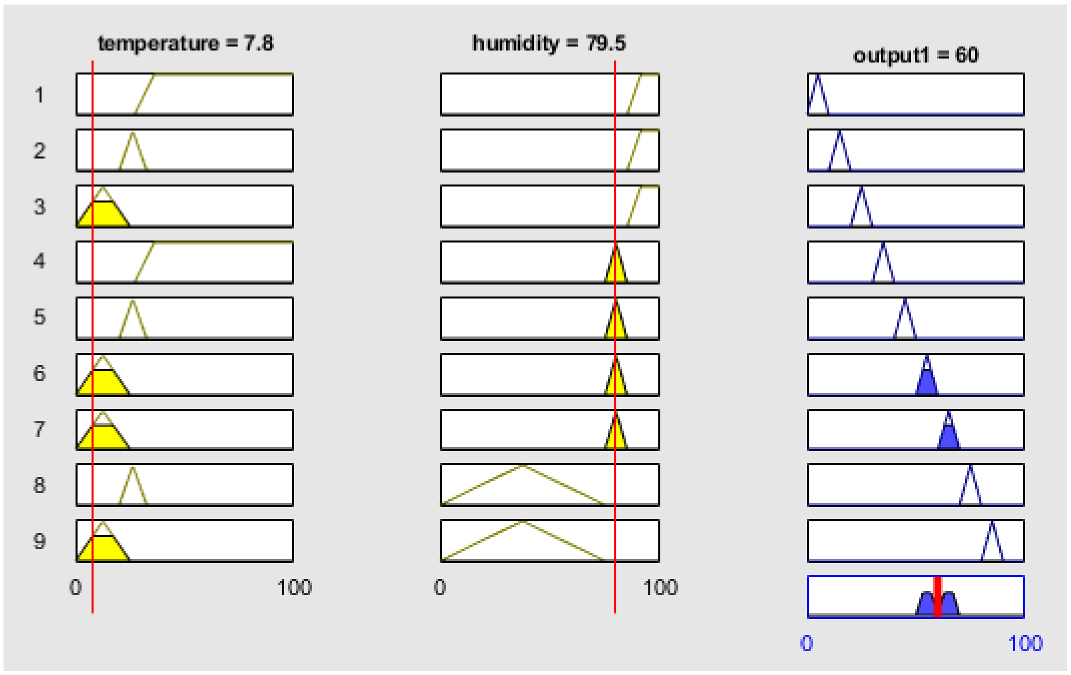


Figure 4. 15: Output results when rule 6 was invoked (temperature and humidity fuzzy logic analysis)

For rule 7 to be invoked, the input temperature had to be of any value between the range of 27.5℃ to 100℃ (This was “hot” membership function for temperature input) while the input humidity had to be of any value in the range of 0% to 75% (This was “low” membership function for humidity input). Figure 4.16 shows the output results at these inputs. It could be seen that the output of 75 unit steps (PHASE 7 value of reference output) resulted when the temperature was 30.7℃ (“hot” membership function value of temperature input) and humidity was 33.2% (“low” membership function value of the humidity input). These outcomes satisfied design
rule 7.

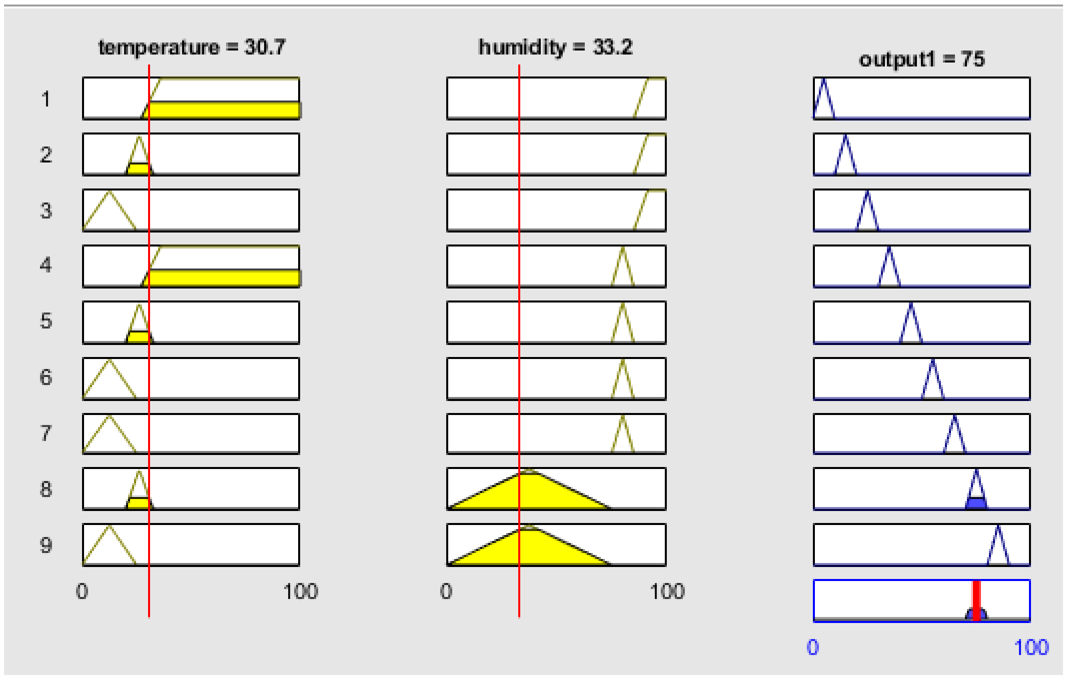


Figure 4. 16: Output results when rule 7 was invoke (temperature and humidity fuzzy logic analysis)

For rule 8 to be invoked, the input temperature had to be of any value in the range of 20℃ to 32.5℃ (This was "normal" membership function for temperature input) while the input humidity had to be of any value in the range of 0% to 75% (This was "low" membership function for humidity input). Figure 4.17 shows the output results at these inputs. It can be seen that the output of 75 unit steps (PHASE 8 value of reference output) resulted when the temperature is 26.1℃ (normal membership function value of temperature input) and humidity was 33.2% ("low" membership function value of the humidity input). These outcomes satisfied design rule 8.

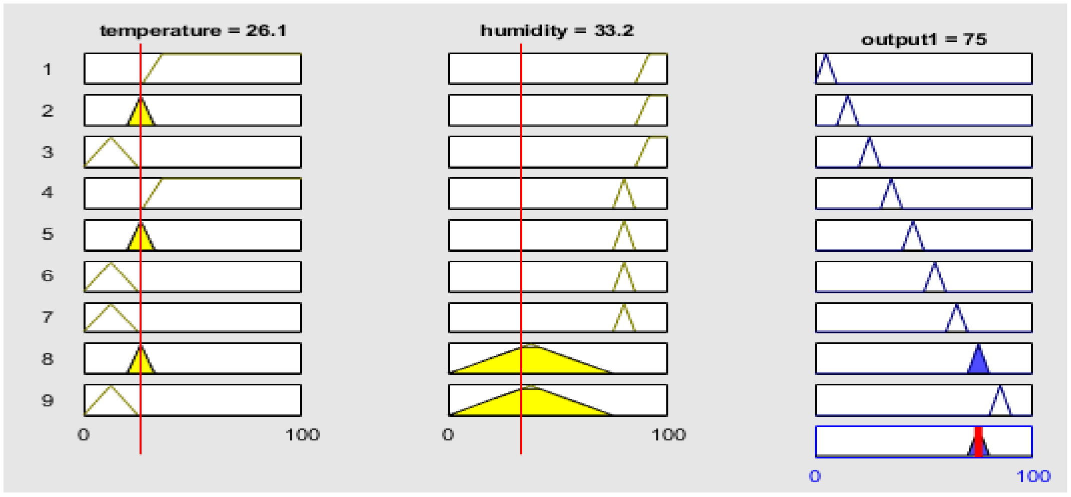


Figure 4. 17: Output results when rule 8 was invoked (temperature and humidity fuzzy logic analysis)

For rule 9 to be invoked, the input temperature had to be of any value in the range of 0℃ to 25℃ (This was “cold” membership function for temperature input) while the input humidity had to be of any value in the range of 0% to 75% (This was “low” membership function for humidity input). Figure 4.18 shows the output results at these inputs. It can be seen that the output of 85 unit steps (PHASE 9 value of reference output) resulted when the temperature was 7.8℃ (cold membership function value of temperature input) and humidity was 33.2% (“low” membership function value of the humidity input). These outcomes satisfied design rule 9.

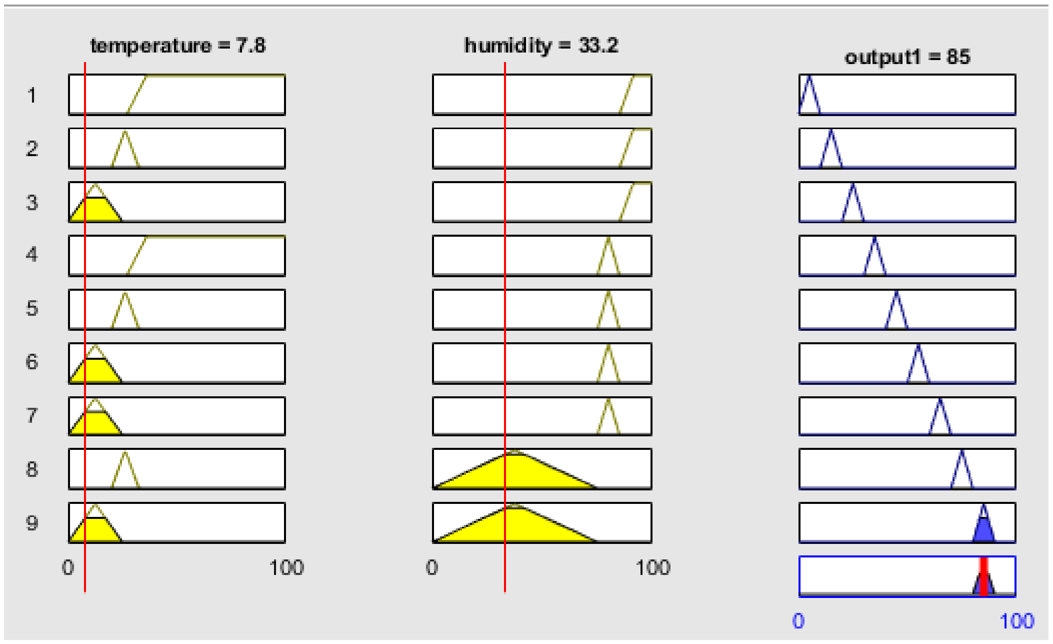


Figure 4. 18: Output results when rule 9 was invoked (temperature and humidity fuzzy logic analysis)

Table 4.7: Arduino output control when rules from rule1 to rule9 are invoked

Humidity/Temp	hot	normal	cold
high	Fan on, heater off, humidifier off [PHASE1] **(rule1)**	Heater off, humidifier off, fan on [PHASE2] **(rule2)**	Heater on, humidifier off, Fan on [PHASE3] **(rule3)**
optimal	Fan on, heater off, humidifier off [PHASE4] **(rule4)**	Heater off, Fan off, humidifier off [PHASE5] **(rule5)**	Humidifier off, heater on, fan off [PHASE6] **(rule6)**
low	Humidifier on, heater off, fan on [PHASE7] **(rule7)**	Heater off, fan off, humidifier on [PHASE8] **(rule8)**	Heater on/humidifier on/fan off (9) [PHASE 9] **(rule9)**

Now that the fuzzy inference rules for fuzzy control of temperature and humidity were designed, Table 4.7 shows how these rules were implemented in the Arduino microcontroller. For instance, if we look at numbered blocks of Table 4.7, it can be seen that when:

i. Fuzzy rule1's output of fuzzy inference is PHASE 1, then the microcontroller automatically switches the fan on, heater off and humidifier off.
ii. Fuzzy rule2's output of fuzzy inference is PHASE 2, then the microcontroller automatically switches the fan on, heater off and humidifier off.
iii. Fuzzy rule3's output of fuzzy inference is PHASE 3, then the microcontroller automatically switches the fan on, heater on and humidifier off.
iv. Fuzzy rule4's output of fuzzy inference is PHASE 4, then the microcontroller automatically switches the fan on, heater off and humidifier off.
v. Fuzzy rule5's output of fuzzy inference is PHASE 5, then the microcontroller automatically switches the fan off, heater off and humidifier off.
vi. Fuzzy rule6's output of fuzzy inference is PHASE 6, then the microcontroller automatically switches the fan off, heater on and humidifier off.
vii. Fuzzy rule7's output of fuzzy inference is PHASE 7, then the microcontroller automatically switches the fan on, heater off and humidifier on.
viii. Fuzzy rule8's output of fuzzy inference is PHASE 8, then the microcontroller automatically switches the fan off, heater off and humidifier on.
ix. Fuzzy rule9's output of fuzzy inference is PHASE 9, then the microcontroller automatically switches the fan off, heater on and humidifier on.

4.6.2 Fuzzy logic results of carbon monoxide (CO) gas

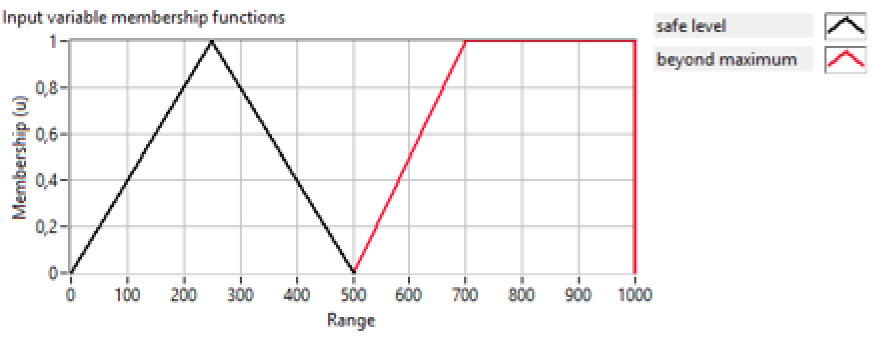


Figure 4. 19: CO input variable with "safe level" and "beyond maximum" membership functions.

The CO input variable is represented as analog steps on the x axis of Figure 4.19. The Arduino microcontroller has 1023 steps of analog readings through its analog inputs. The system only needed 1000 steps to represent CO analog readings as inputs to gas sensor of which a range of 0 to 500 analog steps (black triangle) were readings representing a "safe level" membership function. The red trapezoid represented a "beyond maximum" membership function with a range of 501 to 1000 analog steps. The output variable "CO_2 injector" had two membership functions called "CO_2 injection on" and "CO_2 injection off" as shown in Figure 4.20. These output membership functions were used as destination of fuzzy control rules designed to control CO gas concentration. The designed fuzzy logic rules to control CO gas concentration were as follows:

1. If CO GAS is SAFE LEVEL, then CO_2 INJECTOR is CO_2 INJECTOR OFF
2. If CO GAS is BEYOND MAXIMUM, then CO_2 INJECTOR is CO_2 INJECTOR ON

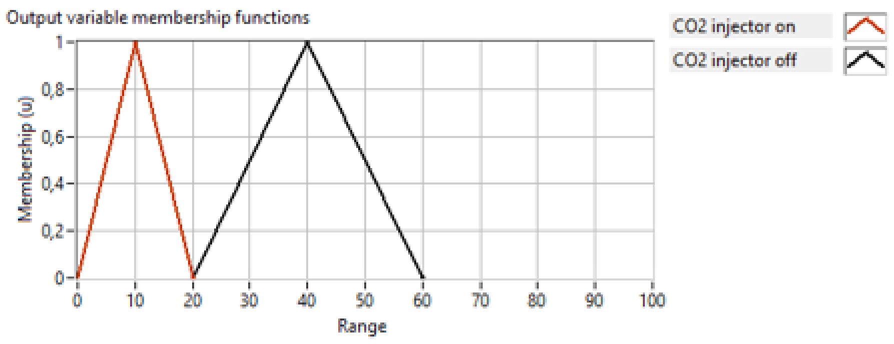


Figure 4. 20: Output variable called CO_2 injector with "CO_2 injector on" and "CO_2 injector off" membership functions.

Surface view Z-output of carbon monoxide is shown by figure 4.21.

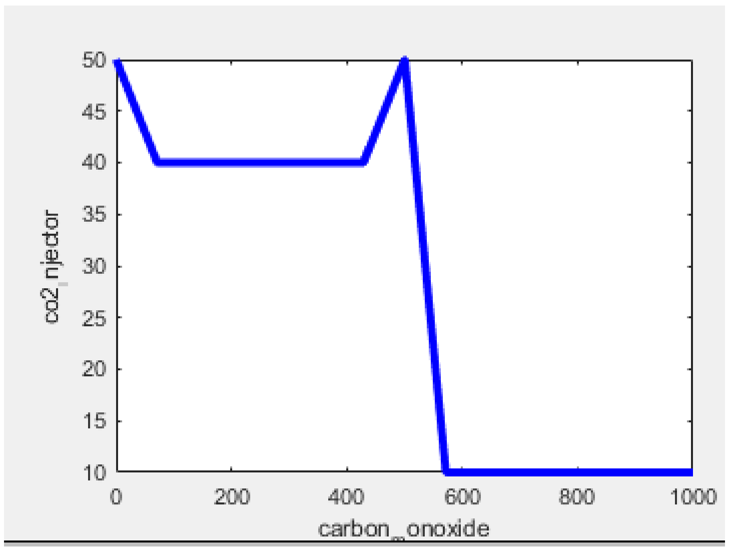


Figure 4. 21: Surface view Z-output of carbon monoxide input x-axis

For rule 1 to be invoked, the CO gas concentration had to be of any value in the range of 0 to 500 analog steps (This was "safe level" membership function for CO input variable). Figure 4.22 shows the output results at this input membership function. It can be seen that the output of 40 unit steps ("CO_2 injector off" value of reference output) resulted when the CO gas concentration was 204 ("safe level" membership function value of CO input variable). These outcomes satisfied design rule 1.

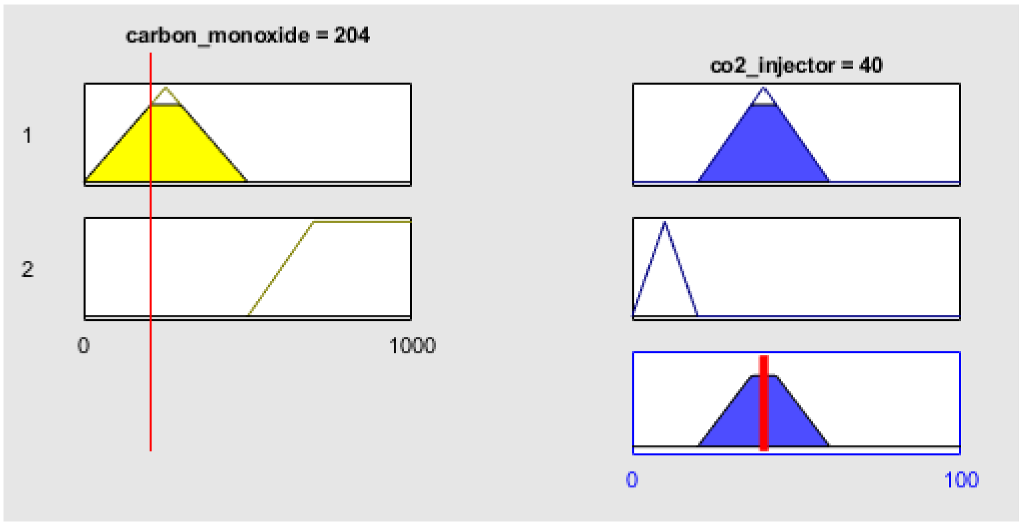


Figure 4. 22: Output results when rule 1 was invoked (CO gas fuzzy logic analysis)

For rule 2 to be invoked, the CO gas concentration had to be of any value in the range of 501 to 1000 analog step (This was "beyond maximum" membership function for CO input variable). Figure 4.23 shows the output results at this input. It can be seen that the output of 10 unit steps ("CO_2 injector on" value of reference output) resulted when the CO gas concentration was 704 ("beyond maximum" membership function value of CO input variable). These outcomes satisfied design rule2.

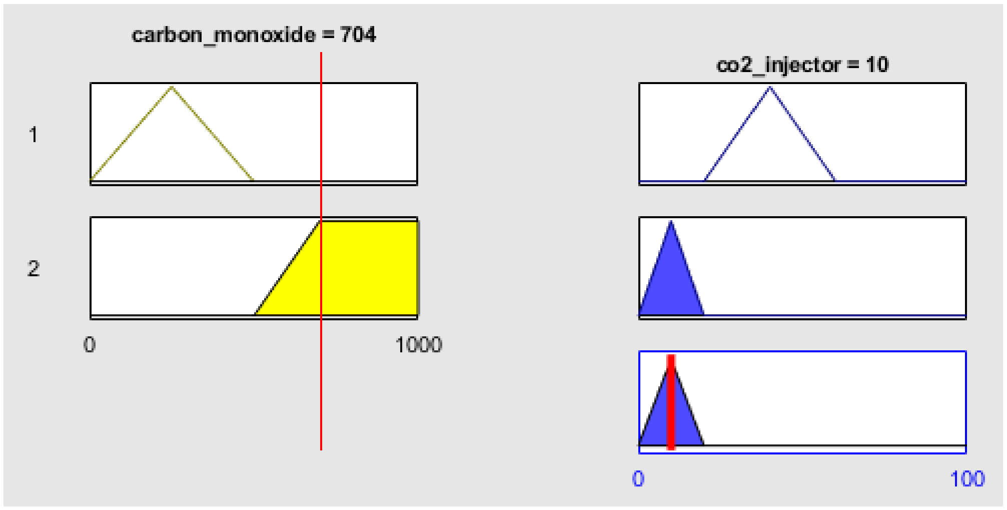


Figure 4. 23: Output results when rule 2 was invoked (CO gas fuzzy logic analysis)

When rule2 of the CO fuzzy inference was invoked, Carbon dioxide (CO_2) gas was injected to neutralize the concentration of Carbon monoxide gas (CO).

4.6.3 Fuzzy logic results of soil moisture content

To detect the degree of soil moisture content, the input variable named soil moisture was used in this study (see Figure 4.24). The maximum analog steps available for this input variable were 1000 steps. This input variable had three membership functions named “dry”, “damp” and “wet” membership functions. The output variable (see Figure 4.25) itself was named “sprinkler” and had two membership functions named “sprinkler on” and “sprinkler off”.

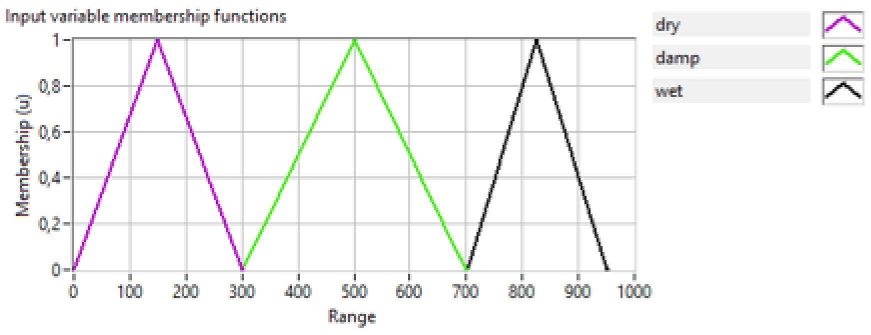


Figure 4. 24: Soil moisture input variable with dry, damp and wet membership functions

The output variable of the soil moisture content is shown in Figure 4.25. The fuzzy inference rules for soil moisture sensor input and its output variable were as shown below.

1. If SOIL MOISTURE is DRY, then SPRINKLER is SPRINKLER ON.
2. If SOIL MOISTURE is DAMP, then SPRINKLER is SPRINKLER OFF.
3. If SOIL MOISTURE is WET, then SPRINKLER is SPRINKLER OFF.

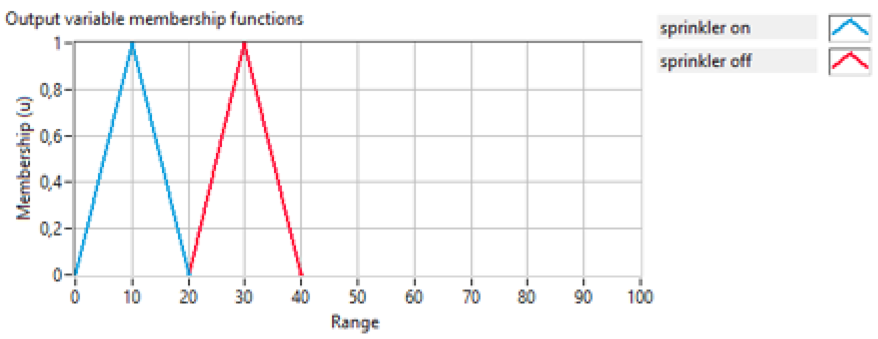


Figure 4. 25: Output variable called sprinkler with sprinkler on and sprinkler off membership functions.

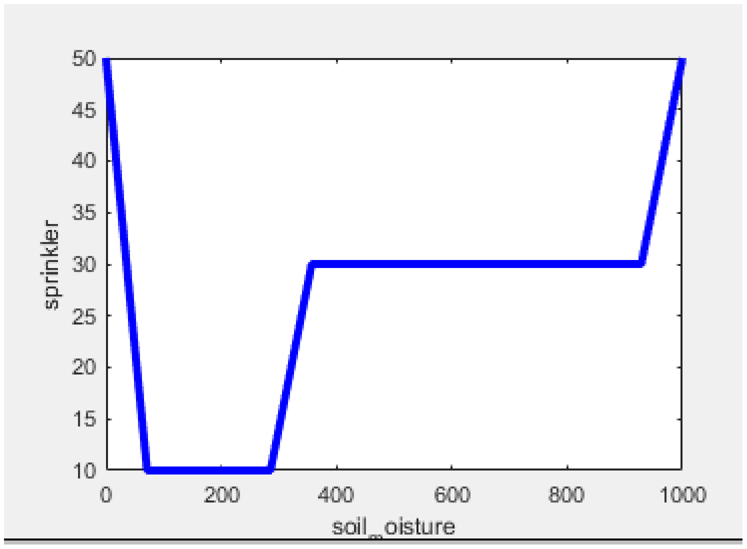


Figure 4. 26: Surface view Z-output of soil moisture input x-axis

For rule 1 to be invoked, the soil moisture had to be of any value in the range of 0 to 300 analog steps (This was “dry” membership function for soil moisture input variable). It can be seen that the output of 10 unit steps (“sprinkler on” value of reference output) resulted when the soil moisture was 101 in the scale (“dry” membership function value of soil moisture input variable). These outcomes satisfied design rule1 as shown in figure 4. 27.

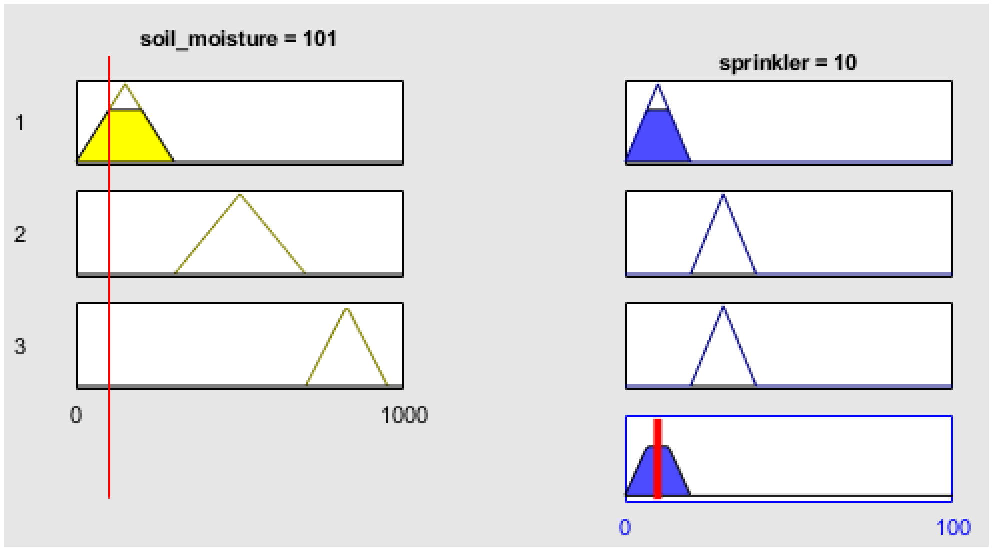


Figure 4. 27: Output result when rule 1 was invoked (soil moisture fuzzy logic analysis)

For rule 2 to be invoked, the soil moisture had to be of any value in the range of 301 to 700 analog steps (This was “damp” membership function for soil moisture input variable). Figure 4.28 shows the output results at this input membership. It can be seen that the output of 30 unit steps (“sprinkler off” value of reference output) resulted when the soil moisture was 503 (“damp” membership function value of soil moisture input variable). These outcomes satisfied design rule2.

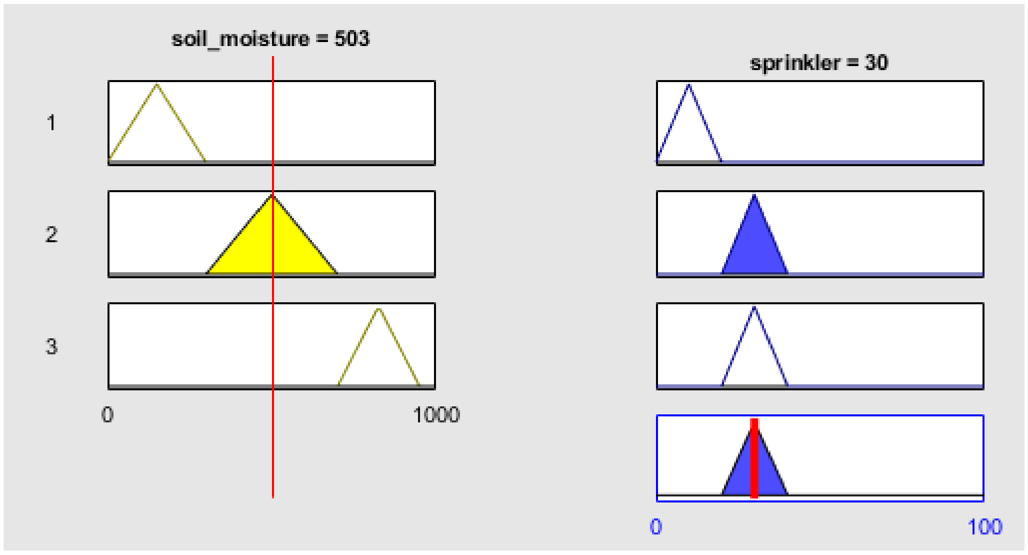


Figure 4. 28: Output results when rule 2 was invoked (soil moisture fuzzy logic analysis)

For rule 3 to be invoked, the soil moisture had to be of any value in the range of 701 to 950 analog steps (This was “wet” membership function for soil moisture input variable). Figure 4.29 shows the output results at this input membership. It can be seen that the output of 30 unit steps (“sprinkler off” value of reference output) resulted when the soil moisture 802 (“wet” membership function value of soil moisture input variable). These outcomes satisfied design rule3.

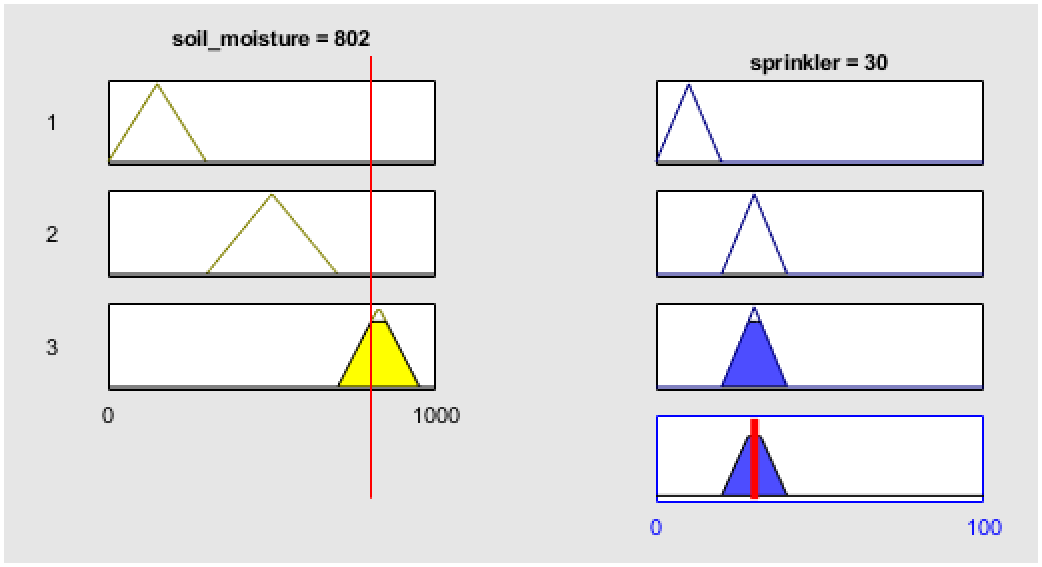


Figure 4. 29: Output result when rule 3 was invoked (soil moisture fuzzy logic analysis)

When rules 2 and 3 were invoked, the microcontroller switched off the sprinkler. When rule 1 was invoked, then the microcontroller switched on the sprinkler.

4.6.4 Fuzzy logic results of light intensity

As shown in figure 4.30, the input variable was named “light intensity” and had two membership functions named “not dark” and “dark”. The maximum analog steps available for this input variable were 1000 steps. The output variable was named “bulb” and had two membership functions named “bulb on” and “bulb off” as shown in figure 4.31.

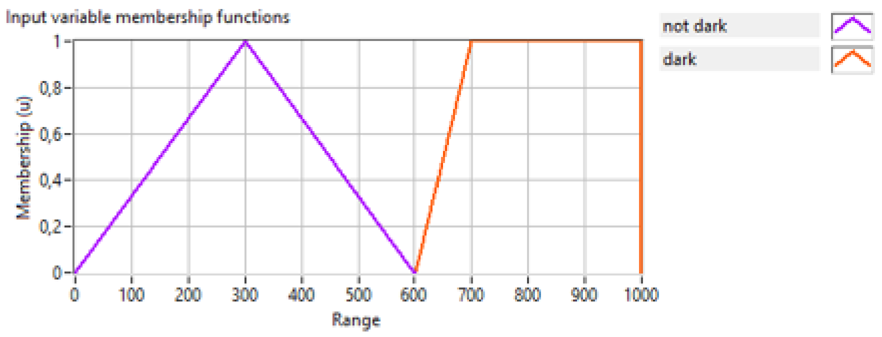


Figure 4. 30: Light intensity input variable with “not dark” and “dark” membership functions

Fuzzy logic design rules for light intensity:

1. If LIGHT INTENSITY is NOT DARK, then BULB is OFF
2. If LIGHT INTENSITY is DARK, then BULB is ON

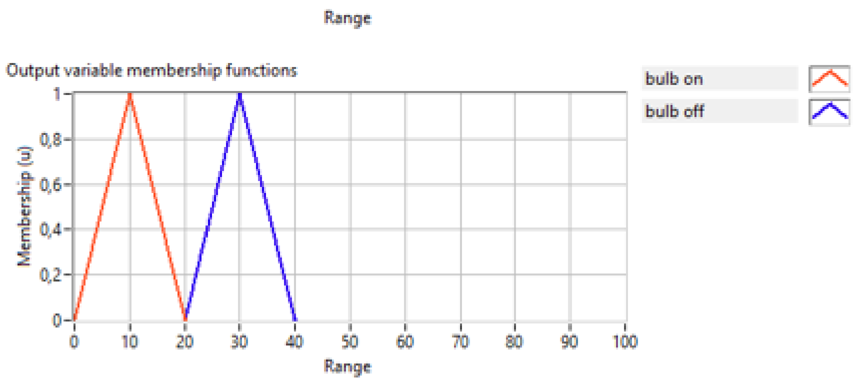


Figure 4. 31: Output variable called bulb and its membership functions

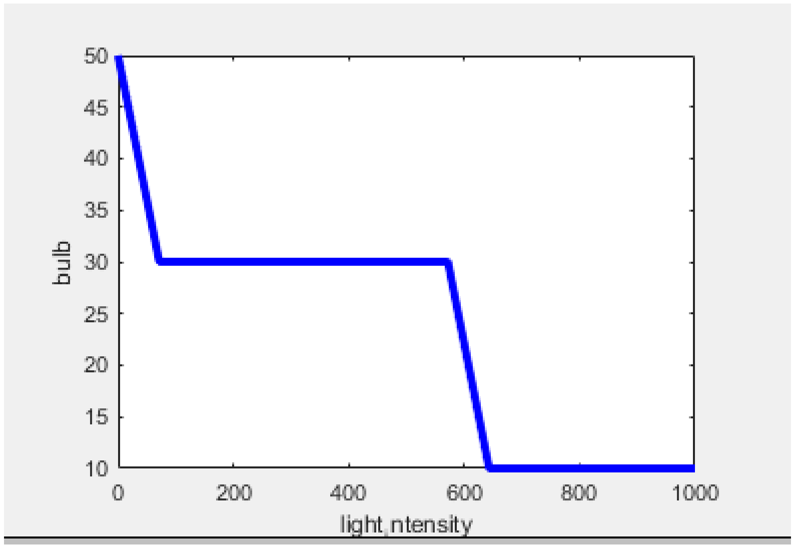


Figure 4. 32: Surface view Z-output of light intensity input x-axis

For rule 1 to be invoked, the "light intensity" had to be of any value in the range of 0 to 600 analog steps (This was "not dark" membership function for soil moisture input variable). Figure 4.33 shows the output results at this input membership. It can be seen that the output of 30 unit steps ("bulb off" value of reference output) resulted when the light intensity 308 ("not dark" membership function value of light intensity input variable). These outcomes satisfied design rule1.

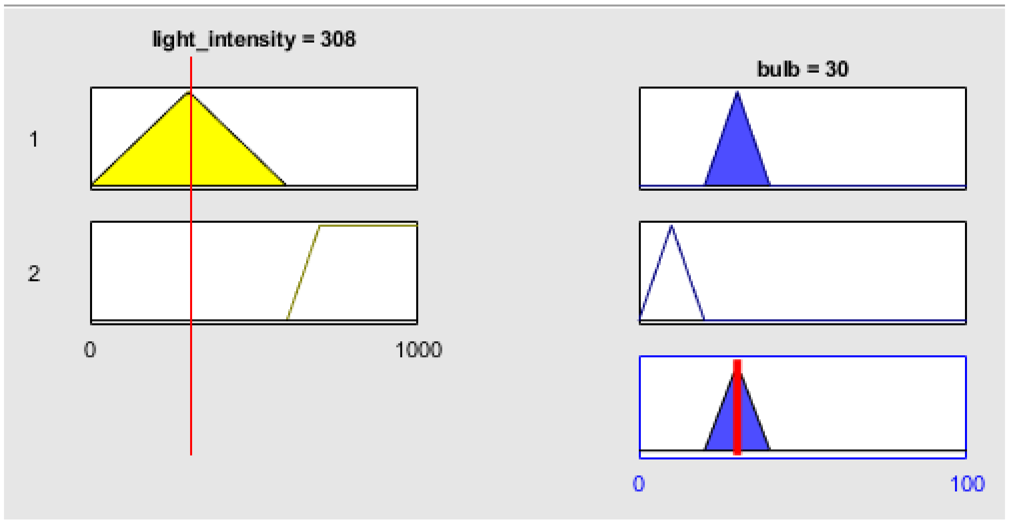


Figure 4. 33: Output results when rule 1 was invoked (light intensity fuzzy logic analysis)

For rule 2 to be invoked, the "light intensity" had to be of any value in the range of 600 to 1000 analog steps (This was "dark" membership function for soil moisture input variable). Figure 4.34 shows the output results at this input membership. It can be seen that the output of 10 unit steps ("bulb on" value of reference output) resulted when the light intensity 662 ("dark" membership function value of light intensity input variable). These outcomes satisfied design rule2.

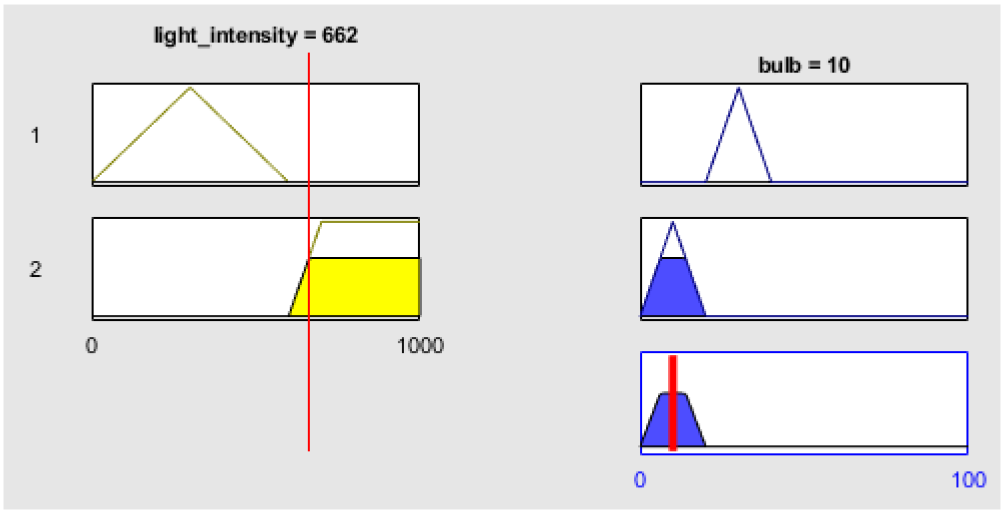


Figure 4. 34: Output results when rule 2 was invoked (light intensity fuzzy logic analysis)

When rule 1 was invoked, then the microcontroller switched off the bulb. When rule2 was invoked, then the microcontroller switched on the bulb.

4.7 Chapter summary

In this chapter we briefly explained the results of the experiments conducted in chapter3.The resulting graphs were analyzed by use accompanying keys and the explanation of each graph was done. The fuzzy logic rules were designed for the fuzzy logic controller of the system and the MATLAB program was used to generate 3D plots for the design rules as well as membership functions of the inputs and outputs. Analysis of the invoked fuzzy logic rules was done in MATLAB and the results matched the design requirements.

CHAPTER 5: Summary of the study, Conclusions, Implications and Recommendations

5.1 Introduction

In the previous chapter, the findings of this study were analyzed with results plotted, tabulated and discussed. In this chapter the summary of the study, conclusions are drawn as well as implications and recommendations are provided. Issues that need further studies are also highlighted.

5.2 Summary of the study

In chapter 1, the study looked at the required design procedure to achieve a system that is capable of monitoring micro climatic parameters of a greenhouse. The intended system was meant to have Arduino microcontrollers capable of interacting with wireless XBee radio modules. The combination of the Arduino micro controllers and wireless radio modules was intended to design three important stages named the sensor station, coordinator station and central station which were meant to interact with one another in order to achieve the desired control goals in the greenhouse. Fuzzy logic control was adopted as means of control over conventional control and Boolean logic control (1/0) because of its ability to design systems without involvement of complex mathematical formulae. Fuzzy logic can be used to mimic the experienced human operator to model the control system. Fuzzy logic is easily embedded in small microcontrollers and processors. Compared with Boolean control(High/Low), fuzzy logic is more accurate. The objectives of the study were also covered in chapter 1. The specific research objectives of this study were as listed below:

i. To determine, design and develop a low-cost microcontroller system for the monitoring of climatic parameters in a greenhouse using smart technology.
ii. The designed system must keep the user consistently informed of the conditions inside a greenhouse.
iii. To use fuzzy logic in order to optimize the control algorithms of a greenhouse monitoring system by use of wireless XBee radio modules and sensors.
iv. To design a system with less installation of complex electric wires in order to save energy.

Acquired sensor data was planned to be displayed on 20X4 LCD installed in the coordinator station of the system. Also, the system was designed such that the user could view sensor data from a remote "central station" by means of a personal computer and a program specially developed using a high level language known as Python.

In chapter 2, previous related works on systems that were designed to monitor the greenhouse environment were reviewed. Market survey of wireless smart technology was also reviewed with accompanying advantages and disadvantages. A comparison of Raspberry Pi board and Arduino microcontroller board was looked at as well as the methods of hooking these boards with a wireless XBee radio module. A review of how the controlled variables could affect the greenhouse environment was also done in chapter 2. A review of sensors recommended for collecting data of the controlled variables was visited in this chapter too. Since wireless XBee radio modules were used in the design of the intended system, it also was important to review routing protocols of wireless communications and how these wireless radios contributed in power management of the systems reviewed in this study.

Chapter 3 was the methodology chapter. The design of the entire research system was conducted in this chapter. This designed research system's three important stations named sensor station, coordinator station and central station were explained in detail. Firstly, we looked at how to program wireless sensors in a program known as XCTU from a company called Digi key. This involved configurations of each of the later mentioned stations and tests for effective communication between coordinator and router XBee radios that were conducted on XCTU terminal windows of the two stations. We then looked at the design of the station known as sensor station by first investigating the sensors involved and then represented the complete circuit diagram of this station with its testing procedures. The LCD that was used to view data in the greenhouse is found in the coordinator station. To change the display when the user wanted to view Carbon monoxide concentrations, a Samsung J3 mobile was used to send Bluetooth commands to the HC-05 Bluetooth module using a Bluetooth terminal available from Google Play Store. The Arduino code for this station was written in order for an Arduino to communicate with the wireless XBee radio connected on top of the Arduino board by means of the wireless proto shield. The Arduino code for the sensor station is included in Appendix A.

Thirdly we looked at how a station known as coordinator station was designed by representing its circuit diagram and testing procedures. The Arduino code was written in order for an Arduino to communicate with the wireless XBee radio connected on top of the Arduino board by means of the wireless proto shield. The fuzzy logic controller code for this station was developed and programmed in the Arduino to coordinate information received from both the sensor station and the central station so that acquired data could be displayed on the LCD and central station. Arduino code of this station is available in Appendix B.

Lastly we looked at tests on how the central station was designed by programming the Arduino with appropriate code so that information could be exchanged between the wireless XBee radio and the Arduino microcontroller. The Arduino code is available in Appendix C. We also looked at how wireless XBee radio was connected on top of the Arduino board by means of the wireless proto shield. A Python code was designed to develop a program for generating sensor data plots. The Python code itself is available in Appendix D. Useful libraries used to enhance all the codes developed in the methodology chapter (Chapter3) were discussed in detail. The experiments covered in this research study were:

i. Experiment 1: Measuring soil moisture content using grove moisture sensor.
ii. Experiment 2: Measuring temperature and humidity using DHT11 sensor.
iii. Experiment 3: Measuring Carbon monoxide content using MQ7 gas sensor.
iv. Experiment 4: Measuring light intensity using LDR.
v. Experiment 5: Demonstrate fuzzy logic control of temperature, humidity, moisture content, Carbon monoxide concentration and light intensity.

Chapter 4 was about the analysis of results. Firstly, we analyzed results of each of the plotted sensor data by tabulating the output results as the graphs changed in value. Since the coordinator station of the designed research system had fuzzy logic involved, MATLAB was used to simulate fuzzy logic controller of the system. The control rules that were designed in MATLAB matched with the recorded results of the system when it was tested. The concept of input and output membership functions was discussed in detail based on MATLAB analysis.

5.3 CONCLUSIONS AS PER RESEARCH OBJECTIVES

5.3.1 General objective

The study set out to investigate the performance, specifications and features of systems that are currently used to monitor greenhouse climatic conditions for Agri-business and the review of studies that were made in effort of improving them. Conclusions in this regard are provided under each research objective.

5.3.2 Research objective1

The first theoretical objective of this study was to carry out literature review in order to determine, design and develop a low-cost microcontroller system for monitoring of climatic parameters in a greenhouse using smart technology.

During the review of the literature pertaining to this study, it was found that cost was the one of the limiting factors that did not encourage agriculturists to use smart agriculture technology for their farming purposes. It was discovered that most of environment monitoring gadgets already in the market like Win Land Enviro Alert system that was reviewed in this study, were not specifically designed for greenhouse climate monitoring but general purpose environment monitoring only.

5.3.3 Research objective2

The second theoretical objective of this study was to design a system that must keep the user consistently informed of the conditions inside a greenhouse.

The greenhouse climate monitoring systems that were reviewed in this study did not have effective means of viewing data. Those that did utilized Microsoft excel which does not support viewing of live streaming data. Some systems like Watchdog wireless crop monitor [43]15 that are already in Agri-business, only have LCD as means of viewing data without any means of remote data viewing.

5.3.4 Research objective3

The third theoretical objective of this study was to use fuzzy logic in order to optimize the control algorithms of a greenhouse monitoring system by use of wireless XBee radio modules and sensors.

The environment monitoring gadgets reviewed in this study have common feature of setting both the humidity and temperature to high or low set points. Hi or Lo settings are crisp values of the controlled input variables. Crisp value is described as either 0 or 1 which means other values within this range are not considered as part of design rules. For this reason, these systems are considered to be inaccurate in terms of data collection and decision making.

5.3.5 Research objective 4

The fourth objective of this study was to design a system with less installation of complex electric wires in order to save energy.

The solution to power management is to design a system with less electrical installations and complex wiring. Some systems that were reviewed in this study involved use of complex electric wires in their design phase with lots of electrical installations and complex wiring.

5.4 Recommendations

The findings and results of this study should be shared with the designing companies that design and manufacture smart technology for climate monitoring in order to enable them to address the problems encountered. Looking at conclusions of objective 1, it is recommended that the Arduino microcontroller be the one used in design of climate monitoring systems as they are cheap and readily available. The Arduino microcontroller boards are also recommended for the on board ADC (analog to digital conversion). This Arduino feature eliminates use of extra ADC circuits that could be costly.

As for conclusions drawn objective 2, it is recommended that remote viewing be included in design stages of the systems that were reviewed in this study. The program to be used for viewing acquired data is recommended to be developed in Python as Python has libraries that enable viewing of live streaming data unlike Microsoft excel.

Fuzzy logic is an approach to computing based on "degrees of truth" rather than the usual "true or false" (1 or 0) Boolean logic on which the modern computer is based. Fuzzy logic includes 0 and 1 as extreme cases of truth (or "the state of matters" or "fact") but also includes the various states of truth in between so that, for example, the result of a comparison between two things could be not "small" or "big" but ".47 of smallness.

With conclusions drawn on objective 3, the recommendation is to design climate monitoring systems that make fuzzy decisions to control the plant which in this study is the greenhouse. With fuzzy logic embedded in the system's microcontroller, the system performance and accuracy are improved as seen in the results of this study.

As for conclusion drawn on objective 4, It is recommended that XBee wireless technology be used in construction of greenhouse monitoring systems as there will be less electrical installation and reduced wiring.

5.5 Limitations of this study

As the system has only one sensor station (sensor node), it is suitable for use in small greenhouses and cannot be used for commercial greenhouses where data is acquired by means of many sensor nodes.

5.6 Future research

As the nature of research always gives rise to other questions, further investigation in the following areas is recommended:

Development of Arduino APIs for fuzzy logic.

Fuzzy logic control was developed by use a library called EFLL [32] in this study. The use of instructions for designing fuzzy logic from the EFLL library [32] is not simple nor straight forward. Therefore, developing the Arduino APIs for fuzzy logic in the Arduino's open source IDE, is the future research area that needs to be investigated. API is a set of functions and procedures that allow the creation of applications which access data of an application.

REFERENCES

[1] L. Bencini, D. Di, G. Collodi, A. Manes, and G. Manes, "Wireless Sensor Networks for On-Field Agricultural Management Process," *Wireless Sensor Networks: Application-Centric Design,* 2010.

[2] Q. Zhang, C.-H. Wu, and K. Tilt, "Application of fuzzy logic in an irrigation control system," *Proceedings of the IEEE International Conference on Industrial Technology (ICIT96).*

[3] X. Wang, W. Liu, L. Gu, C. Sun, C. Gu, and C. D. Silva, "Development of an intelligent control system for wood drying processes," *2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings (Cat. No.01TH8556).*

[4] K. Prema, N. S. Kumar, S. S. Dash, and S. S. Chandran, "Online control of fuzzy based mine detecting robot using virtual instrumentation," *2012 International Conference on Computing, Communication and Applications*, 2012.

[5] H. Mansor, S. B. M. Noor, R. M. K. R. Ahmad, F. S. Taip, and O. F. Lutfi, "Fuzzy Control of Grain Drying Process," *2009 11th International Conference on Computer Modelling and Simulation*, 2009.

[6] "NexSens Technology Inc." [Online]. Available: http://www.bing.com/cr?IG=6C6D1BA35DBE410496A4B81F5DAC857A&CID=39E2915198DC68FE3CD09AF69973698F&rd=1&h=f7IOaJiZmyWNgS7p-7dBvX28RurkM5cqS2PlIPvIi6Y&v=1&r=http%3a%2f%2fwww.nexsens.com%2f&p=DevEx,5069.1. [Accessed: 03-Mar-2017].

[7] N. Sigrimis, P. Antsaklis, and P. Groumpos, "Advances in control of agriculture and the environment," *IEEE Control Systems Magazine*, vol. 21, no. 5, pp. 8–12, 2001.

[8] H. Bremner and B. Postlethwaite, "The development of a relational fuzzy model based controller for an industrial process," *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*

[9] S. N. Singh, R. Jha, and M. K. Nandwana, "Optimal design of solar powered fuzzy control irrigation system for cultivation of green vegetable plants in Rural India," *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*, 2012

[10] A. Abdullah, S. A. Enazi, and I. Damaj, “AgriSys: A smart and ubiquitous controlled-environment agriculture system,” *2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, 2016.

[11] J. Wang and Y. Haiye, “Research on greenhouse intelligent remote control system,” *Proceedings of the 2015 Information Technology and Mechatronics Engineering Conference*, 2015

[12] M. Z. D. Abad and S. K. M. Mashhadi, “Design and implementation for temp-weight and humidity control of dryer based on fuzzy logic,” *The 2nd International Conference on Control, Instrumentation and Automation*, 2011.

[13] “Automated Irrigation System using Wireless Sensor Network and Raspberry Pi,” *International Journal of Science and Research (IJSR)*, vol. 4, no. 12, pp. 2056–2058, May 2015.

[14] R. E. King and N. Sigrimis, “Computational intelligence in crop production,” *Computers and Electronics in Agriculture*, vol. 31, no. 1, pp. 1–3, 2001.

[15]”Absoluteautomation,“WatchdogCropMonitor”.[Online].Available:http://www.absoluteautomation.com/crop_monitor/. [Accessed on: 24-Apr-2017]

[16] Routing protocols for wireless networks. [Online]. Available:https://en.wikipedia.org/wiki/Routing_protocol.[Accessed:27-Jun-2017]”

[17] Narasimhan.L.V, Arvind. A, Bever.K, “Greenhouse asset management using wireless sensor actor networks”, *In Proceedings of IEEE International Conference on Mobile Ubiquitous*, pp.1-47,2007

[18] Meng-Shiuan Pan and Yu-Chee Tseng, “ZigBee Wireless Sensor Networks and Their Applications”, National Chiao Tung University, 2013

[19] V. Jelicic, D. Tolic, and V. Bilas, “Consensus-based decentralized resource sharing between co-located Wireless Sensor Networks,” *2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 2014.

[20] Mukhopadhyay , Ed , *Smart Sensing Technology for Agriculture and Environmental Monitoring*, LNEE 146, pp.21-32, 2012

[21] Mukhopadhyay , Ed , *Smart Sensing Technology for Agriculture and Environmental Monitoring*, LNEE 146, pp.82-102, 2012

[22] J. Wang, X. Niu, L. Zheng, C. Zheng, and Y. Wang, “Wireless Mid-Infrared Spectroscopy Sensor Network for Automatic Carbon Dioxide Fertilization in a Greenhouse Environment,” *Sensors*, vol. 16, no. 12, p. 1941, 2016.

[23] Mukhopadhyay , Ed , *Smart Sensing Technology for Agriculture and Environmental Monitoring*, LNEE 146, pp.187-200, 2012

[24] A. R. Al-Ali, M *et al* "ZigBee-based irrigation system for home gardens”, *ICCSPA*, vol.8, no.31, pp. 1-5, 2015.

[25] Salazar, “Temperature Control of a Thermal Plasma Torch with Inductive Coupling Using the Arduino Board”, Department of Computer Engineering and Automation, Federal University of Rio Grande do Norte, Brasil, vol.4, pp.499-504,2014

[26] WIRELESS TECHNOLOGIES IN PROCESS AUTOMATION - REVIEW AND ...” [Online]. Available: http://www.bing.com/cr?IG. [Accessed: 08-Mar-2017].

[27] XBee WiFi-Hookup [Online]. Available:https://learn.sparkfun.com/tutorials/xbee-wifi-hookup-guide/using-x-ctu [Accessed:03-Jul-2017]”

[28] Configure your modules. [Online].

Available: http://docs.digi.com/display/XCTU/Configure+your+modules

[29] Bakker, J.C., Greenhouse climate control: an integrated approach 1st Ed, J.C. Bakker, Wageningen Press, 1995.

[30] Python Organization.[Online].Available: https://www.python.org/

[Accessed:04-Apr-2017]

[31] DHT sensor library. [Online]. Available: https://github.com/adafruit/DHT-sensor-library [Accessed:06-Feb-2017]

[32] H. Rhoades, “Why Do Plants Grow with Light: How Light Affects Plants,” *Gardening Know How*, 28-Mar-2015. [Online]. Available: http://www.gardeningknowhow.com/problems/how-light-affects-the-growth-of-a-plant-problems-with-too-little-light.htm. [Accessed: 03-Mar-2018].

[33] Levanon, D.M. B. Macham, “Organic materials degradation for CO_2 enrichment of greenhouse crops”. Carbon dioxide enrichment of greenhouse crops, 1st Ed, New York: CRC Press, 1986, pp. 123-145.

[34] H. Bremner and B. Postlethwaite, “The development of a relational fuzzy model based controller for an industrial process,” *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*.

[35] A. Rahali, M. Guerbaoui, A. Ed-Dahhak, Y. E. Afou, A. Tannouche, A. Lachhab, and B. Bouchikhi, “Development of a data acquisition and greenhouse control system based on GSM,” *International Journal of Engineering, Science and Technology*, vol. 3, no. 8, 2012.

[36] Sigrimis. N, King. R, “Special Issue on Intelligent Systems in Crop Production” *Computers and electronics in agriculture*, vol.31, pp.1-105, 2000.

[37] Sigrimis.N, King.R, “Special Issue on Advances in greenhouse environment control”, *Computers and electronics in agriculture*, vol.26, pp.217-374, 1999.

[38] J. Pickin, S. Yuen, and H. Hennings, “Waste management options to reduce greenhouse gas emissions from paper in Australia,” *Atmospheric Environment*, vol. 36, no. 4, pp. 741–752, 2002.

[39] A. Dresch, D. P. Lacerda, and J. A. V. A. Jr, “Design Science Research,” 2015.

[40] *Seattle Robotics Society*. [Online]. Available: http://www.seattlerobotics.org/encoder/mar98/fuz/fl_part1.html. [Accessed: 23-Feb-2017].

[41] Ahmad Ibrahim, *Fuzzy logic for embedded systems applications 3rd edition*, New York, Prentice Hall

[42] Timothy J Ross, *Fuzzy logic with engineering applications 1st edition*, New York, Prentice Hall

[43] T. Point, “Artificial Intelligence Fuzzy Logic Systems,” *www.tutorialspoint.com*, 08-Jan-2017. [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_fuzzy_logic_systems.htm. [Accessed: 24-Feb-2017]

[44] H. Yap, “Faculty of 1000 evaluation for Anthropogenic ocean acidification over the twenty-first century and its impact on calcifying organisms.,” *F1000 - Post-publication peer review of the biomedical literature*, Apr. 2005.

[45] T. C. Weiler, “The Greenhouse Environment. The Effect of Environmental Factors on the Growth and Development of Flower Crops.John W. Mastalerz,” *The Quarterly Review of Biology*, vol. 53, no. 3, pp. 323–323, 1978.

[46] Nelson, P.V., Greenhouse operation and management 6 Ed, Upper Saddle River, Prentice Hall, 2003.

[47] Hopkins, W.G., Plant development, Chelsea House Publishers, 1995, pp 151.

[48] L. Ortho and O. Books, All about Greenhouses, New York, John Wiley and Sons, 2001.

[49] J. J. Hanan, W. D. Holley, and K. L. Goldsberry, “Greenhouse Management,” *Advanced Series in Agricultural Sciences*, 1978.

[50] Nelson, K.S, “Greenhouse management for flower and plant production 2 Ed”, Danville: Interstate Printers & Publishers, 1980.

[51] G. Barrenetxea, F. Ingelrest, G. Schaefer, and M. Vetterli, “Wireless Sensor Networks for Environmental Monitoring: The SensorScope Experience,” *2008 IEEE International Zurich Seminar on Communications*, 2008.

[52] V. Raghavan and H. Shahnasser, “Embedded Wireless Sensor Network for Environment Monitoring,” *Journal of Advances in Computer Networks*, vol. 3, no. 1, pp. 13–17, 2015.

[53] S.U.Zagade, Prof Kawitkar, “Wireless sensor network in greenhouse”, *International Journal of Engineering Research and Technology (IJERT)*, vol.1, no4, pp.1-3, 2012

[54] D. Chaudhary, S. Nayse, and L. Waghmare, “Application of Wireless Sensor Networks for Greenhouse Parameter Control in Precision Agriculture,” *International Journal of Wireless & Mobile Networks*, vol. 3, no. 1, pp. 140–149, 2011.

[55] “Tutorial 38: Use an Arduino as a Slave with Python (Nanpy),” *The Zan Show*. [Online]. Available: http://thezanshow.com/electronics-tutorials/raspberry-pi/tutorial-38. [Accessed: 27-Apr-2017].

[56] “EnviroAlert EA800-ip - Winland.” [Online]. Available: http://www.absoluteautomation.com/documents/usr/winland/ea800_ownersmanual.pdf

[Accessed: 04-Apr-2017].

[57] Katemopoulos, M. “The History of the Greenhouse”. [Online]: Available: http://sharonfalsetto.suite101.com/history-of-the-greenhouse-a81808 .[Accessed:03-Mar-2017]

[58] Robert Faludi, Building wireless sensor networks “A practical guide to the ZigBee mesh networking protocol”, New York City, O’Reilly ,2010

[59] M. Lieschnegg, A. Fuchs, B. Lechner, and O. Mariani, “Autonomous Sensor Platform for Environmental Monitoring Applications,” *Lecture Notes in Electrical Engineering Smart Sensing Technology for Agriculture and Environmental Monitoring*, pp. 187–200, 2012.

[60] A. Dubendorf, “Wireless Data Technologies”, New York City, John Wiley and sons,2003

[61] Mukhopadhyay, Ed, *Smart Sensing Technology for Agriculture and*

Environmental Monitoring, LNEE 146, pp.1-20, 2012

[62] E.Elahi and G.Schwender, “Introduction to the ZigBee Wireless Sensor and Control Network”, Prentice hall, 2009

[63] M. M. Hasan and K. Arshad, “Robust home automation scheme using cognitive ZigBee network,” *Ict 2013*, 2013.

[Accessed:03-Jul-2017]

[64] Arduino fuzzy logic library.[Online].Available: https://github.com/zerokol/eFLL [Accessed:03-Mar-2017]

APPENDIX A

```
#include <dht.h>

#define dht_apin A0 // Analog Pin sensor is connected to Dht11

#define ldr_pin A3

#define MoistsensorPin A1 //Analog Pin sensor is connected to Moisture sensor

dht DHT;

int readValue=0;

const int AOUTpin=A2;//the AOUT pin of the CO sensor goes into analog pin A0 of the
arduino

const int DOUTpin=8;//the DOUT pin of the CO sensor goes into digital pin D8 of the arduino

void setup(){

Serial.begin(9600);

pinMode(DOUTpin, OUTPUT);//sets the pin as an input to the arduino

  }

  void loop(){

// Temperature and humidity

DHT.read11(dht_apin);

delay(1000);

float t = DHT.temperature;

float h = DHT.humidity;

 // read the value from the sensors:

int moistsensorValue = analogRead(MoistsensorPin); // Moisture sensor

int ldrValue = analogRead(ldr_pin); // Moisture sensor
int gasvalue= analogRead(AOUTpin);//reads the analaog value from the CO sensor's AOUT
pin
Serial.print("CO value: ");
Serial.println(gasvalue);//prints the CO value
delay(250);
Serial.print("<M");
Serial.print(moistsensorValue);
Serial.print(">");
delay(250);
Serial.print("<T");
Serial.print(t);
Serial.print(">");
delay(250);
Serial.print("<H");
Serial.print(h);
Serial.print(">");
delay(250);
if (gasvalue > 500){
Serial.print("<G");
Serial.print(gasvalue);
Serial.print(">");
delay(250);
 }
else{
```

```
Serial.print("<N");

Serial.print(gasvalue);

Serial.print(">");

delay(250);

 }

Serial.print("<L");

Serial.print(ldrValue);

Serial.print(">");

delay(250);    }
```

APPENDIX B

```
#include <LiquidCrystal.h>

#include <FuzzyRule.h>

#include <FuzzyComposition.h>

#include <Fuzzy.h>

#include <FuzzyRuleConsequent.h>

#include <FuzzyOutput.h>

#include <FuzzyInput.h>

#include <FuzzyIO.h>

#include <FuzzySet.h>

#include <FuzzyRuleAntecedent.h>

#include <SoftwareSerial.h>

#define heaterpin 6

#define humidpin 7

#define fanpin 8

#define sprinklerpin 50

#define lightrpin 51


LiquidCrystal lcd(12, 9, 5, 4, 3, 2);

SoftwareSerial BTserial(10, 11); // RX | TX

Fuzzy* fuzzy = new Fuzzy();// Instantiating an object library

int temperature; // Declaring temperature variable

int humidity;// Declaring humidity variable

int soilmoisture;// Declaring soilmoisture variable
int gasvalue;

int light;

int discard;

int state;

char Datain[24];

byte list;

boolean began = false;

boolean dead = false;

char a;


void setup()

{

lcd.begin(20,4);

// Print a message to the LCD.

lcd.setCursor(4,0);

lcd.print("FUZZY LOGIC");

lcd.setCursor(1,1);

lcd.print("GREEN HOUSE CLIMATE");

lcd.setCursor(3,2);

lcd.print("MONITOR SYSTEM");

delay(5000);

BTserial.begin(9600); // BLUETOOTH SET UP

Serial.begin(9600);

pinMode(13,OUTPUT);
pinMode(humidpin,OUTPUT);

pinMode(heaterpin,OUTPUT);

pinMode(fanpin,OUTPUT);

pinMode(sprinklerpin,OUTPUT);

pinMode(lightpin,OUTPUT);

// Creating Fuzzy input temperature input 1

FuzzyInput* temperature = new FuzzyInput(1);// Creating a FuzzyInput temperature

FuzzySet* cold = new FuzzySet(0, 0, 17, 25);

temperature->addFuzzySet(cold); // Add FuzzySet cold to FuzzyInput temperature

FuzzySet* normal = new FuzzySet(20, 26.5,26.5, 32.5);

temperature->addFuzzySet(normal); // Add FuzzySet normal to FuzzyInput temperature

FuzzySet* hot = new FuzzySet(27.5, 35.5,100,100);

temperature->addFuzzySet(hot); // Add FuzzySet hot to FuzzyInput temperature

fuzzy->addFuzzyInput(temperature); // Add FuzzyInput to Fuzzy object

// Creating Fuzzy input humidity input2

FuzzyInput* humidity = new FuzzyInput(2);// Creating a FuzzyInput temperature

FuzzySet* low = new FuzzySet(0, 37,37, 75);

humidity->addFuzzySet(low); // Add FuzzySet low to FuzzyInput temperature

FuzzySet* optimal = new FuzzySet( 75, 80,80, 85);

humidity->addFuzzySet(optimal); // Add FuzzySet optimal to FuzzyInput temperature

FuzzySet* high = new FuzzySet(85, 90, 100, 100);

humidity->addFuzzySet(high); // Add FuzzySet high to FuzzyInput temperature

fuzzy->addFuzzyInput(humidity); // Add FuzzyInput to Fuzzy object

// Creating FuzzyOutput control for temperature and humidity
```

```
  FuzzyOutput* control = new FuzzyOutput(1);
  FuzzySet* phase1 = new FuzzySet(0, 5, 5, 10);
  control->addFuzzySet(phase1); // Add FuzzySet phase1 to control
  FuzzySet* phase2 = new FuzzySet(10, 15, 15, 20);
  control->addFuzzySet(phase2); // Add FuzzySet phase2 to control
  FuzzySet* phase3 = new FuzzySet(20, 25, 25, 30);
  control->addFuzzySet(phase3); // Add FuzzySet phase3 to control
  FuzzySet* phase4 = new FuzzySet(30, 35, 35, 40);
  control->addFuzzySet(phase4); // Add FuzzySet phase4 to control
  FuzzySet* phase5 = new FuzzySet(40, 45, 45, 50);
  control->addFuzzySet(phase5); // Add FuzzySet phase5 to control
  FuzzySet* phase6 = new FuzzySet(50, 55, 55, 60);
  control->addFuzzySet(phase6); // Add FuzzySet phase6 to control
  FuzzySet* phase7 = new FuzzySet(60, 65, 65, 70);
  control->addFuzzySet(phase7); // Add FuzzySet phase7 to control
  FuzzySet* phase8 = new FuzzySet(70, 75, 75, 80);
  control->addFuzzySet(phase8); // Add FuzzySet phase8 to control
  FuzzySet* phase9 = new FuzzySet(80, 85, 85, 90);
  control->addFuzzySet(phase9); // Add FuzzySet phase9 to control
  fuzzy->addFuzzyOutput(control); // Add FuzzyOutput to Fuzzy object
// Creating rules for temperature and humidity from rule 1 to rule 9
// Rule1
FuzzyRuleAntecedent* ifTemperatureHotAndhumidityHigh = new FuzzyRuleAntecedent();
ifTemperatureHotAndhumidityHigh->joinWithAND(hot, high);
```

```
FuzzyRuleConsequent* thenControlphase1 = new FuzzyRuleConsequent(); // Instantiating a
Consequent to expression

thenControlphase1->addOutput(phase1);// Adding corresponding FuzzySet to Consequent
object

// Instantiating a FuzzyRule object

FuzzyRule* fuzzyRule01 = new FuzzyRule(1,
ifTemperatureHotAndhumidityHigh,thenControlphase1 );

fuzzy->addFuzzyRule(fuzzyRule01); // Adding FuzzyRule to Fuzzy object

// Rule2

FuzzyRuleAntecedent* ifTemperatureNormalAndhumidityHigh = new
FuzzyRuleAntecedent();

ifTemperatureNormalAndhumidityHigh->joinWithAND(normal, high);

FuzzyRuleConsequent* thenControlphase2 = new FuzzyRuleConsequent(); // Instantiating a
Consequent to expression

thenControlphase2->addOutput(phase2);// Adding corresponding FuzzySet to Consequent
object

// Instantiating a FuzzyRule object

FuzzyRule* fuzzyRule02 = new FuzzyRule(2,
ifTemperatureNormalAndhumidityHigh,thenControlphase2 );

fuzzy->addFuzzyRule(fuzzyRule02); // Adding FuzzyRule to Fuzzy object

// Rule3

FuzzyRuleAntecedent* ifTemperatureColdAndhumidityHigh = new FuzzyRuleAntecedent();

ifTemperatureColdAndhumidityHigh->joinWithAND(cold, high);

FuzzyRuleConsequent* thenControlphase3 = new FuzzyRuleConsequent(); // Instantiating a
Consequent to expression
```

```
thenControlphase3->addOutput(phase3);// Adding corresponding FuzzySet to Consequent
object

// Instantiating a FuzzyRule object

FuzzyRule* fuzzyRule03 = new FuzzyRule(3,
ifTemperatureColdAndhumidityHigh,thenControlphase3 );

fuzzy->addFuzzyRule(fuzzyRule03); // Adding FuzzyRule to Fuzzy object

// Rule4

FuzzyRuleAntecedent* ifTemperatureHotAndhumidityOptimal = new FuzzyRuleAntecedent();

ifTemperatureHotAndhumidityOptimal->joinWithAND(hot, optimal);

FuzzyRuleConsequent* thenControlphase4 = new FuzzyRuleConsequent(); // Instantiating a
Consequent to expression

thenControlphase4->addOutput(phase4);// Adding corresponding FuzzySet to Consequent
object

// Instantiating a FuzzyRule object

FuzzyRule* fuzzyRule04 = new FuzzyRule(4,
ifTemperatureHotAndhumidityOptimal,thenControlphase4 );

fuzzy->addFuzzyRule(fuzzyRule04); // Adding FuzzyRule to Fuzzy object

// Rule5

FuzzyRuleAntecedent* ifTemperatureNormalAndhumidityOptimal = new
FuzzyRuleAntecedent();

ifTemperatureNormalAndhumidityOptimal->joinWithAND(normal, optimal);

FuzzyRuleConsequent* thenControlphase5 = new FuzzyRuleConsequent(); // Instantiating a
Consequent to expression

thenControlphase5->addOutput(phase5);// Adding corresponding FuzzySet to Consequent
object

// Instantiating a FuzzyRule object
```

```
FuzzyRule* fuzzyRule05 = new FuzzyRule(5,
ifTemperatureNormalAndhumidityOptimal,thenControlphase5 );

fuzzy->addFuzzyRule(fuzzyRule05); // Adding FuzzyRule to Fuzzy object

// Rule6

FuzzyRuleAntecedent* ifTemperatureColdAndhumidityOptimal = new
FuzzyRuleAntecedent();

ifTemperatureColdAndhumidityOptimal->joinWithAND(cold, optimal);

FuzzyRuleConsequent* thenControlphase6 = new FuzzyRuleConsequent(); // Instantiating a
Consequent to expression

thenControlphase6->addOutput(phase6);// Adding corresponding FuzzySet to Consequent
object

// Instantiating a FuzzyRule object

FuzzyRule* fuzzyRule06 = new FuzzyRule(6,
ifTemperatureColdAndhumidityOptimal,thenControlphase6 );

fuzzy->addFuzzyRule(fuzzyRule06); // Adding FuzzyRule to Fuzzy object

// Rule7

FuzzyRuleAntecedent* ifTemperatureHotAndhumidityLow = new FuzzyRuleAntecedent();

ifTemperatureHotAndhumidityLow->joinWithAND(hot, low);

FuzzyRuleConsequent* thenControlphase7 = new FuzzyRuleConsequent(); // Instantiating a
Consequent to expression

thenControlphase7->addOutput(phase7);// Adding corresponding FuzzySet to Consequent
object

// Instantiating a FuzzyRule object

FuzzyRule* fuzzyRule07 = new FuzzyRule(7,
ifTemperatureHotAndhumidityLow,thenControlphase7 );

fuzzy->addFuzzyRule(fuzzyRule07); // Adding FuzzyRule to Fuzzy object
```

```
// Rule8
FuzzyRuleAntecedent* ifTemperatureNormalAndhumidityLow = new FuzzyRuleAntecedent();
ifTemperatureNormalAndhumidityLow->joinWithAND(normal, low);
FuzzyRuleConsequent* thenControlphase8 = new FuzzyRuleConsequent(); // Instantiating a
Consequent to expression
thenControlphase8->addOutput(phase8);// Adding corresponding FuzzySet to Consequent
object
// Instantiating a FuzzyRule object
FuzzyRule* fuzzyRule08 = new FuzzyRule(8,
ifTemperatureNormalAndhumidityLow,thenControlphase8 );
fuzzy->addFuzzyRule(fuzzyRule08); // Adding FuzzyRule to Fuzzy object
// Rule9
FuzzyRuleAntecedent* ifTemperatureColdAndhumidityLow = new FuzzyRuleAntecedent();
ifTemperatureColdAndhumidityLow->joinWithAND(cold, low);
FuzzyRuleConsequent* thenControlphase9 = new FuzzyRuleConsequent(); // Instantiating a
Consequent to expression
thenControlphase9->addOutput(phase9);// Adding corresponding FuzzySet to Consequent
object
// Instantiating a FuzzyRule object
FuzzyRule* fuzzyRule09 = new FuzzyRule(9,
ifTemperatureColdAndhumidityLow,thenControlphase9 );
fuzzy->addFuzzyRule(fuzzyRule09); // Adding FuzzyRule to Fuzzy object
//Creating rules for moisture sensor
delay(5000);
lcd.clear();
```

```
}

void loop()

{

while(Serial.available() > 0)

{

//Start recieving data

char aDhar = Serial.read();

if(aDhar == '<')

{

began = true;

list = 0;

Datain[list] = '\0';

}

else if(aDhar == '>')

{

dead= true;

}

else if(began)

{

Datain[list] = aDhar;

list++;

Datain[list] = ',';

}

}
```

```
if(began && dead)
{
// Use the value
//Read temperature
if(Datain[0] == 'T')
{
Datain[0] = ' ';
int temperature = atoi(Datain);
Serial.print("<T");
Serial.print(temperature);
Serial.print(">");
lcd.setCursor(0,0);
lcd.print("TEMPERATURE=");
lcd.print(temperature);
lcd.print(" ");
lcd.print("DEGR");
delay(10);
//BTserial.print(temperature);
//BTserial.print(",");
fuzzy->setInput(1, temperature); //Set inputs.in paranthesis is ID then input name
  }
//Read humidity
else if(Datain[0] == 'H')
{
```

```
Datain[0] = ' ';

int humidity = atoi(Datain);

Serial.print("<H");

Serial.print(humidity);

Serial.print(">");

delay(10);

lcd.setCursor(3,1);

lcd.print("HUMIDITY=");

lcd.print(humidity);

delay(10);

lcd.print(" ");

lcd.print("%");

//BTserial.print(humidity);

//BTserial.print(",");

fuzzy->setInput(2, humidity); //Set inputs.in paranthesis is ID then input name

}

//Read moisture content

else if(Datain[0] == 'M')

{

Datain[0] = ' ';

int soilmoisture = atoi(Datain);

Serial.print("<M");

Serial.print(soilmoisture);

Serial.print(">");
```

```
delay(10);

if (soilmoisture>=0&&soilmoisture<=400){  // Fuzzification of soilmoisture when input is dry

  digitalWrite(sprinklerpin,HIGH);  // Control rule after fuzzification

  }

else if (soilmoisture>400&&soilmoisture<=950){ // Fuzzification of soilmoisture when input is
wet

  digitalWrite(sprinklerpin,LOW); // Control rule after fuzzification

  }

lcd.setCursor(0,2);

lcd.print("SOILMOISTURE=");

lcd.print(soilmoisture);

lcd.print(" ");

lcd.print("mc");

lcd.print("   ");

//BTserial.print(soilmoisture);

//BTserial.print(",");

delay(10);

}

//Read gas above treshold and show presence of CO

else if(Datain[0] == 'G'){

Datain[0] = ' ';

int gasvalue = atoi(inData);

Serial.print("<G");

Serial.print(gasvalue);
Serial.print(">");

delay(10);

while(BTserial.available()>0){

//write on lcd that there is CO gas

state = BTserial.read();

if(state== '0'){

lcd.clear();

lcd.setCursor(0,0);

lcd.print("CO GAS DETECTED");

lcd.setCursor(0,1);

lcd.print("CO LEVEL=");

lcd.print(gasvalue);

lcd.print(" ");

lcd.print("ppm");

lcd.print(" ");

lcd.setCursor(0,2);

lcd.print("WARNING!: REDUCE CO");

delay(5000);

lcd.clear();

}

else{

break;

  }

}
```

```
  }
 //Read gas above treshold and show absence of CO
else if(Datain[0] == 'N')
{
Datain[0] = ' ';
int gasvalue = atoi(inData);
Serial.print("<N");
Serial.print(gasvalue);
Serial.print(">");
delay(10);
while(BTserial.available()>0){
//write on lcd that there is no CO gas
state = BTserial.read();
if(state=='1'){
lcd.clear();
lcd.setCursor(0,0);
lcd.print("####################");
lcd.setCursor(1,1);
lcd.print("AIR GAS CONTENT IS");
lcd.setCursor(9,2);
lcd.print("OK");
lcd.setCursor(0,3);
lcd.print("####################");
delay(5000);
lcd.clear();}
else {
  break;
  }


  }
  }
//Read Light intensity
else if(inData[0] == 'L')
{
Datain[0] = ' ';
int light = atoi(Datain);
if (light>=0&&light<=600){  // Fuzzification of light intensisty
  digitalWrite(lightpin,LOW);  // Control rule after fuzzification
  }
else if (light>600&&light<=1023){ // Fuzzification of light
  digitalWrite(lightpin,HIGH); // Control rule after fuzzification
  }
Serial.print("<L");
Serial.print(light);
Serial.print(">");
delay(10);
lcd.setCursor(4,3);
lcd.print("LIGHT=");
lcd.print(light);
delay(10);
lcd.print(" ");
lcd.print(" lx");
}
// Blink Led to show that data is recieved from the remote xBee
digitalWrite(13,HIGH);
delay(100);
digitalWrite(13,LOW);
delay(100);
// Enter the fuzzy logic code
fuzzy->fuzzify();              // Then fuzzify
float output1 = fuzzy->defuzzify(1);    // defuzzify if possiple
if (fuzzy->isFiredRule(1)==true){         // Check fired rule
  digitalWrite(fanpin,HIGH);
  digitalWrite(heaterpin,LOW);
  digitalWrite(humidpin,LOW);
  }
if (fuzzy->isFiredRule(2)==true){         // Check fired rule
  digitalWrite(fanpin,HIGH);
  digitalWrite(heaterpin,LOW);
  digitalWrite(humidpin,LOW);
  }
 if (fuzzy->isFiredRule(3)==true){         // Check fired rule
```

```
  digitalWrite(fanpin,HIGH);
  digitalWrite(heaterpin,HIGH);
  digitalWrite(humidpin,LOW);
  }
if (fuzzy->isFiredRule(4)==true){        // Check fired rule
  digitalWrite(fanpin,LOW);
  digitalWrite(heaterpin,HIGH);
  digitalWrite(humidpin,LOW);
  }
if (fuzzy->isFiredRule(5)==true){        // Check fired rule
  digitalWrite(fanpin,LOW);
  digitalWrite(heaterpin,LOW);
  digitalWrite(humidpin,LOW);
  }
if (fuzzy->isFiredRule(6)==true){        // Check fired rule
  digitalWrite(fanpin,LOW);
  digitalWrite(heaterpin,HIGH);
  digitalWrite(humidpin,LOW);
  }
if (fuzzy->isFiredRule(7)==true){        // Check fired rule
  digitalWrite(fanpin,HIGH);
  digitalWrite (heaterpin, LOW);
  digitalWrite (humidpin, HIGH);
  }
```

```
if (fuzzy->isFiredRule(8)==true){          // Check fired rule
  digitalWrite(fanpin,LOW);
  digitalWrite(heaterpin,LOW);
  digitalWrite(humidpin,HIGH);
  }
if (fuzzy->isFiredRule(9)==true){          // Check fired rule
  digitalWrite (fanpin, LOW);
  digitalWrite(heaterpin,HIGH);
  digitalWrite(humidpin,HIGH);
  }
began = false;
dead = false;
list = 0;
Datain[index] = '\0';
  }
}
```

APPENDIX C

```
int temperature; // Declaring temperature variable

int humidity;// Declaring humidity variable

int soilmoisture;// Declaring soilmoisture variable

int gasvalue;

int light;

int discard;

int state;

char Datain[24];

byte list;

boolean began = false;

boolean dead = false;

char a;

void setup(){

 Serial.begin(9600);

  }

void loop()

{

while(Serial.available() > 0)

{

//Start recieving data

char aDhar = Serial.read();

if(aDhar == '<')

{
```

```
began = true;

list = 0;

Datain[index] = '\0';

}

else if(aDhar == '>')

{

dead = true;

}

else if(began)

{

Datain[list] = aDhar;

list++;

Datain[list] = ',';

}

}

if(began && dead)

{

// Use the value

//Read temperature

if(iDatain[0] == 'T')

{

Datain[0] = ' ';

int temperature = atoi(Datain);

Serial.print(temperature);
Serial.print(",");
  }
//Read humidity
else if(Datain[0] == 'H')
{
inData[0] = ' ';
int humidity = atoi(Datain);
Serial.print(humidity);
Serial.print(",");
  }
//Read moisture sensor
else if(Datain[0] == 'M')
{
Datain[0] = ' ';
int soilmoisture = atoi(Datain);
Serial.print(soilmoisture);
Serial.print(",");
}
//Read gas presence
else if(Datain[0] == 'G')
{
Datain[0] = ' ';
 gasvalue = atoi(inData);
Serial.print(gasvalue);
Serial.print(",");
}
//Read gas absence
else if(Datain[0] == 'N')
{
Datain[0] = ' ';
 gasvalue = atoi(Datain);
Serial.print(gasvalue);
Serial.print(",");
}
//Read Light intensity
else if(Datain[0] == 'L')
{
Datain[0] = ' ';
int light = atoi(Datain);
Serial.println(light);
}
started = false;
dead = false;
list= 0;
Datain[list] = '\0';
}
}
```

APPENDIX D

```
import numpy # imports numpy library

import serial # imports serial library

import matplotlib.pyplot  as plt #imports matplolib

import time

from drawnow import*

tempDegrees = []

humidityPercent = []

soilmoisturePercent =[]

gasPPM = []

lightLux = []

arduinoData = serial.Serial('COM3',9600) #

plt.ion() #

cnt = 0

def makeFig():

   plt.title('Live streaming Sensor data')

   plt.grid(True)

   plt.ylabel('Temperature Degr.celcius/Humidity%/CO Gas.PPM/Light intensity.Lux/Soil 
moisture.mc')

   plt.plot(tempDegrees,'g-*',label='Temperature Degr.celcius')

   plt.plot(humidityPercent,'r-.',label='Humidity%')

   plt.plot(soilmoisturePercent,'b-.',label='Soil moisture.mc')

   plt.plot(gasPPM,'v-.',label='CO Gas.ppm' )

   plt.plot(lightLux,'y-.',label='Light intensity.lx ' )
```

```
    plt.legend(loc='upper right')
while True: #while loop that loops forever
    while (arduinoData.inWaiting()==0): #Wait here untilthere is data
        pass #do nothing
    arduinoString = arduinoData.readline() #read the line of text from an array
    pass #do nothing
    arduinoString = arduinoData.readline() #read the line of text from an array
    dataArray = arduinoString.split(',') #Split the line of text into an array called dataArray
    temperature = int (dataArray[1])
    humidity = int (dataArray[2])
    soilmoisture = int (dataArray[0])
    gasvalue = int (dataArray[3])
    light = int (dataArray[4])
    tempDegrees.append(temperature)
    humidityPercent.append(humidity)
    soilmoisturePercent.append(soilmoisture)
    gasPPM.append(gasvalue)
    lightLux.append(light)
    drawnow(makeFig) #Call drawnow to update our live graph
    plt.pause(.00001)
    cnt=cnt+1
    if(cnt>50) :
        tempDegrees.pop(0)
        humidityPercent.pop(0)
```

```
gasPPM.pop(0)

soilmoisturePercent.pop(0)

lightLux.pop(0)
```