1. INTRODUCTION

The temperature control system maintains the system's temperature as per the requirement of the system. It takes the environment's current temperature as an input for this purpose. With the help of these values, temperature of the system is either heated or cooled as per the desired value.

The heating and cooling mechanisms can be controlled using various techniques like Proportional Control mechanisms, Proportional-Integral-Derivative (PID) control, Cascade Control, Model Predictive Control, Fuzzy Logic, etc. This project will use Fuzzy Logic as a control mechanism to operate the cooling mechanism. Fuzzy Logic is a probabilistic methodology which gives an approximate value. Fuzzy Logic's approach to control problem mimics how a person would make decisions, only much faster.

1.1 Problem Definition

The aim of the FL temperature control system is to control the temperature of the system in relation with the surrounding temperature using. It uses the temperature of the system sensed by a temperature sensor. Then for maintaining temperature which is predefined to be at temperature sensor reading of 60, fuzzification and defuzzification principles are applied. The project that we implemented shows whether the surrounding is more or less than the threshold value. Two LED's are used to depict whether the surrounding is hot (red) or cold (blue).

For this implementation, we use an Arduino Microcontroller, LED's for showing the output and LM35 temperature sensor to sense the current temperature of the surrounding.

1.2 Scope of the Topic

This project can be used to maintain the temperature of the system at 15.55 Degree Celsius which is predefined. The accuracy of this system depends upon the accuracy of the temperature sensor used (LM35).

1.3 Relevance of the Project

Temperature control system can be used in various huge applications like air conditioners which needs appropriate humidity as well as in washing machines. It is also used in laboratories which need fixed temperatures.

2. Description

2.1 Fuzzy Logic (FL)

The concept of Fuzzy Logic was put forward by Lotfi Zadeh, a professor at the University of California at Berkley. Fuzzy Logic is a form of many-valued logic or probabilistic logic. It deals with reasoning that is approximate and not exact or accurate. Entire Fuzzy Logic is based upon fuzzy logic variables. These variables can have any value ranging in between 0 (extreme false) and 1 (extreme true). Professor Zadeh reasoned that people do not require precise, numerical information input, and yet they are capable of highly adaptive control. FL is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems.

FL incorporates a simple, rule-based **IF X AND Y THEN Z** approach to solve a control problem rather than an attempt to model a system mathematically. The FL model is empirically-based, relying on an operator's experience rather than their technical understanding of the system.

For example, rather than dealing with temperature control in terms such as "SP =500F", "T <1000F", or "210C <TEMP <220C", terms like "IF (process is too cool) AND (process is getting colder) THEN (add heat to the process)" or "IF (process is too hot) AND (process is heating rapidly) THEN (cool the process quickly)" are used. These terms are imprecise and yet very descriptive of what must actually happen. Consider what a person does in the shower if the temperature is too cold: one will make the water comfortable very quickly with little trouble. FL is capable of mimicking this type of behaviour but at very high rate.

FL is used because:

i. It is inherently robust since it does not require precise, noise-free inputs and can be programmed to fail safely if a feedback sensor quits or is destroyed. The output control is a smooth control function despite a wide range of input variations.

ii. Since the FL controller processes user-defined rules governing the target control system, it can be modified and tweaked easily to improve or drastically alter system performance. New sensors can easily be incorporated into the system simply by generating appropriate governing rules.

iii. FL can control nonlinear systems that would be difficult or impossible to model mathematically. This opens doors for control systems that would normally be deemed unfeasible for automation.

iv. FL is not limited to a few feedback inputs and one or two control outputs, nor is it necessary to measure or compute rate-of-change parameters in order for it to be implemented.

v. Any sensor data that provides some indication of a system's actions and reactions is sufficient. This allows the sensors to be inexpensive and imprecise thus keeping the overall system cost and complexity low.

2.2 Working pattern of Fuzzy Logic

i. Define clearly the control objectives (What is to be controlled?) and the criteria (What should be done to control the system?).
ii. Then, decide upon the input as well as output relationships and choose a minimum number of parameters or fuzzy logic variables as inputs which are in general the significant error and the significant rate-of-change-of- error.
iii. Using the rule-based structure of FL, break the control problem down into a series of IF X AND Y THEN Z rules that define the desired system output response for given system input conditions. The number and complexity of rules depends on the number of input parameters that are to be processed and the number fuzzy variables associated with each parameter.
iv. Create FL membership functions that define the meaning (values) of Input / Output terms used in the rules.
v. Program the rules into the FL hardware engine.
vi. Test the system, evaluate the results, tune the rules and membership functions, and retest until satisfactory results are obtained.

2.3 Linguistic Variables

In 1973, Professor Lotfi Zadeh proposed the concept of linguistic or "fuzzy" variables. Think of them as linguistic objects or words, rather than numbers. The sensor input is a noun, e.g. "temperature", "displacement", "velocity", "flow", "pressure", etc. Since error is just the difference, it can be thought of the same way. The fuzzy variables themselves are adjectives that modify the variable (e.g. "large positive" error, "small positive" error, "zero" error, "small negative" error, and "large negative" error). As a minimum, one could simply have "positive", "zero", and "negative" variables for each of the parameters. Additional ranges such as "very large" and "very small" could also be added to extend the responsiveness to exceptional or very nonlinear conditions, but aren't necessary in a basic system.

FL does not require precise inputs, is inherently robust, and can process any reasonable number of inputs but system complexity increases rapidly with more inputs and outputs. Distributed processors would probably be easier to implement. Simple, plain-language IF X AND Y THEN Z rules are used to describe the desired system response in terms of linguistic variables rather than mathematical formulas.

The number of these is dependent on the number of inputs, outputs, and the designer's control response goals.

Linguistic variables are more or less known as fuzzy logic variables which act as inputs.

2.4 Rule Matrix

The fuzzy parameters used are the adjectives negative, zero, and positive. To picture this, imagine the simplest practical implementation, a 3-by-3 matrix. The columns represent negative error, zero error, and positive error inputs from left to right. The rows represent negative, zero, and positive error-dot input from top to bottom. This planar construct is called a rule matrix. It has two input conditions, error and error-dot, and one output response conclusion (at the intersection of each row and column). In this case there are nine possible logical products (AND) output response conclusions.

Although not absolutely necessary, rule matrices usually have an odd number of rows and columns to accommodate a zero centre row and column region. This may not be needed as long as the functions on either side of the centre overlap somewhat and continuous dithering of the output is acceptable since the zero regions correspond to no change output responses the lack of this region will cause the system to continually hunt for zero. It is also possible to have a different number of rows than columns. This occurs when numerous degrees of inputs are needed. The maximum number of possible rules is simply the product of the number of rows and columns, but definition of all of these rules may not be necessary since some input conditions may never occur in practical operation. The primary objective of this construct is to map out the universe of possible inputs while keeping the system sufficiently under control.

Linguistic variables are used to represent an FL system's operating parameters. The rule matrix is a simple graphical tool for mapping the FL control system rules. It accommodates two input variables and expresses their logical product (AND) as one output response variable. To use, define the system using plain-English rules based upon the inputs, decide appropriate output response conclusions, and load these into the rule matrix.

2.5 Process of Implementation

The first step in implementing FL is to decide exactly what is to be controlled and how. For example, suppose we want to design a simple proportional temperature controller with an electric heating element and a variable-speed cooling fan. A positive signal output calls for 0-100 % heat while a negative signal output calls for 0-

100 % cooling. Control is achieved through proper balance and control of these two active devices.

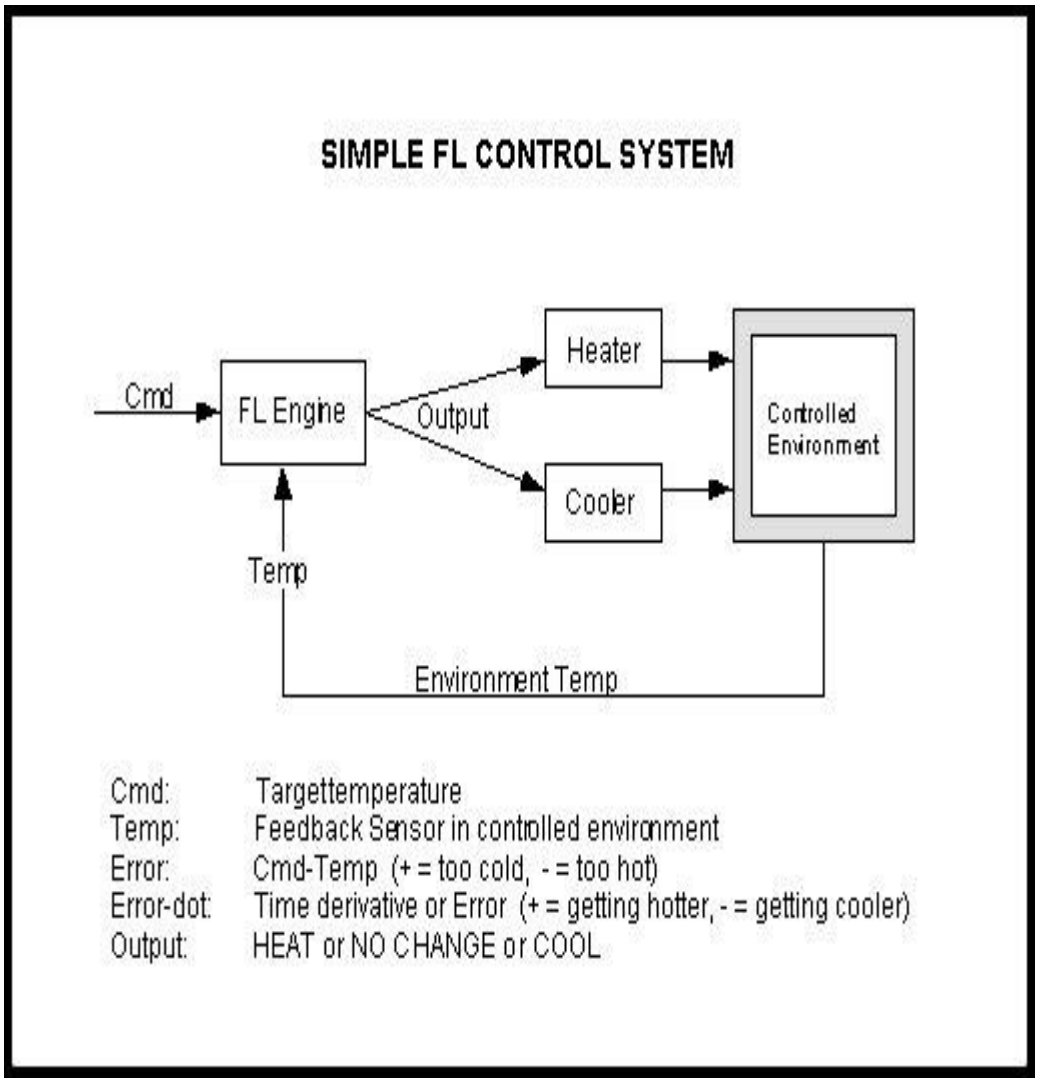


Fig 2.1: A simple block diagram of the control system.

It is necessary to establish a meaningful system for representing the linguistic variables in the matrix. For this example, the following will be used:
"N" = "negative" error or error-dot input level
"Z" = "zero" error or error-dot input level
"P" = "positive" error or error-dot input level
"H" = "Heat" output response
"-" = "No Change" to current output
"C" = "Cool" output response

Define the minimum number of possible input product combinations and corresponding output response conclusions using these terms. For a three-by-three matrix with heating and cooling output responses, all nine rules will need to be defined. The conclusions to the rules with the linguistic variables associated with the output response for each rule are transferred to the matrix.

2.5.1 System Operating Rules

Linguistic rules describing the control system consist of two parts; an antecedent block (between the IF and THEN) and a consequent block (following THEN). By making this type of evaluation, usually done by an experienced operator, fewer rules can be evaluated, thus simplifying the processing logic and perhaps even improving the system performance.

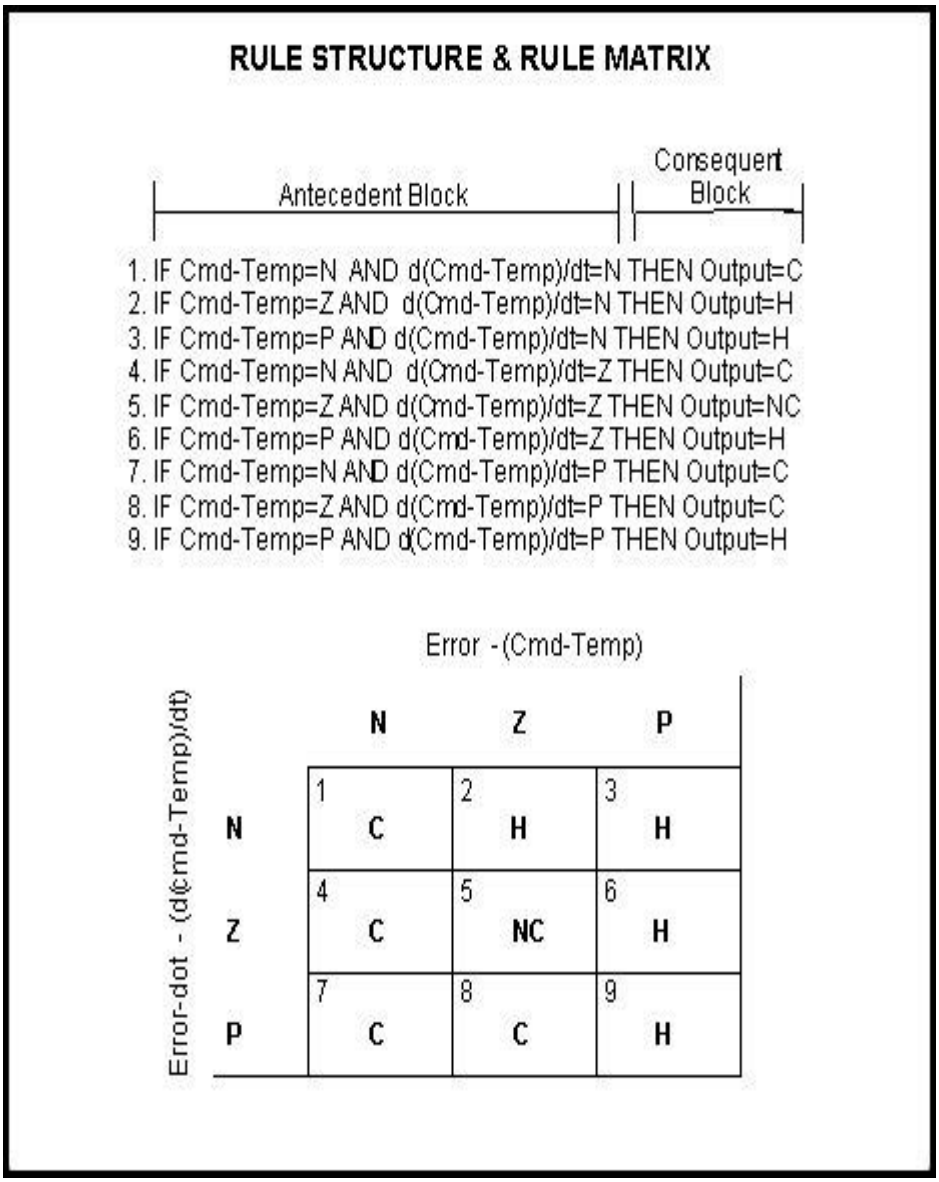


Fig 2.2: The Rule Structure

After transferring the conclusions from the nine rules to the matrix there is a noticeable symmetry to the matrix. This suggests (but doesn't guarantee) a reasonably well-behaved (linear) system. This implementation may prove to be too simplistic for some control problems; however it does illustrate the process. This will increase the rule base size and complexity but may also increase the quality of the control. Figure 2 shows the rule matrix derived from the previous rules.

2.6 Membership Functions

The membership function is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processed, define functional overlap between inputs, and ultimately determines an output response. The rules use the input membership values as weighting factors to determine their influence on the fuzzy output sets of the final output conclusion. Once the functions are inferred, scaled, and combined, they are defuzzified into a crisp output which drives the system. There are different membership functions associated with each input and output response. Some features to note are:

SHAPE - triangular is common, but bell, trapezoidal, haversine and, exponential have been used. More complex functions are possible but require greater computing overhead to implement. HEIGHT or magnitude (usually normalized to 1) WIDTH (of the base of function), SHOULDERING (locks height at maximum if an outer function. Shouldered functions evaluate as 1.0 past their centre) CENTRE points (centre of the member function shape) OVERLAP (N&Z, Z&P, typically about 50% of width but can be less).

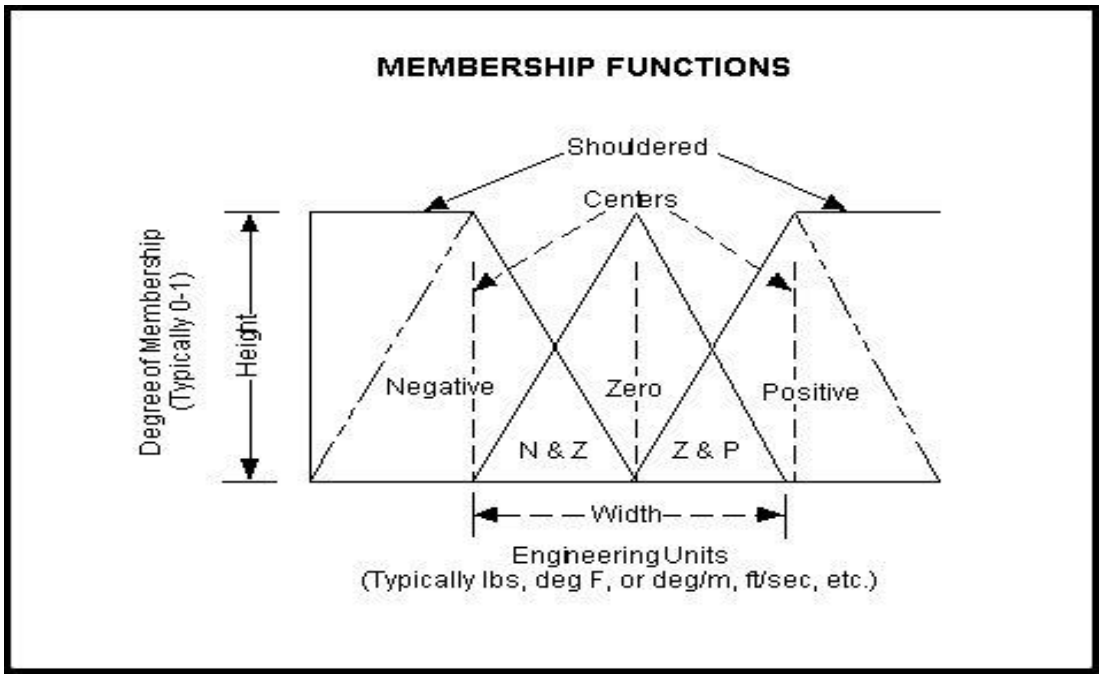


Fig 2.3: Features of the membership function

Figure 2.3 illustrates the features of the triangular membership function which is used in this example because of its mathematical simplicity. Other shapes can be used but the triangular shape lends itself to this illustration.

The degree of membership (DOM) is determined by plugging the selected input parameter (error or error-dot) into the horizontal axis and projecting vertically to the upper boundary of the membership function(s).

2.6.1 Error & Error-Dot Function Membership

The degree of membership for an "error" of -1.0 projects up to the middle of the overlapping part of the "negative" and "zero" function so the result is "negative" membership = 0.5 and "zero" membership = 0.5. Only rules associated with "negative" & "zero" error will actually apply to the output response. This selects only the left and middle columns of the rule matrix.

For an "error-dot" of +2.5, a "zero" and "positive" membership of 0.5 is indicated. This selects the middle and bottom rows of the rule matrix. By overlaying the two regions of the rule matrix, it can be seen that only the rules in the 2-by-2 square in the lower left corner (rules 4,5,7,8) of the rules matrix will generate non-zero output conclusions. The others have a zero weighting due to the logical AND in the rules.

As inputs are received by the system, the rule base is evaluated. The antecedent (IF X AND Y) blocks test the inputs and produce conclusions. The consequent (THEN Z) blocks of some rules are satisfied while others are not. The conclusions are combined to form logical sums. These conclusions feed into the inference process where each response output member function's firing strength (0 to 1) is determined.

INPUT DEGREE OF MEMBERSHIP

Error = -1.0: negative = 0.5 and zero = 0.5
Error-dot = +2.5: zero = 0.5 and positive = 0.5
ANTECEDENT & CONSEQUENT BLOCKS (e = error, er = error-dot or error-rate)

Now referring back to the rules, plug in the membership function weights from above. Error selects rules 1,2,4,5,7,8 while error-dot selects rules 4 through 9. Error and error-dot for all rules are combined to a logical product (LP or AND, that is the minimum of either term). Of the nine rules selected, only four (rules 4, 5, 7 and 8) fire or have non-zero results. This leaves fuzzy output response magnitudes for only Cooling and No Change which must be inferred, combined, and defuzzified to return the actual crisp output. In the rule list below, the following definitions apply: (e) =error, (er) =error-dot.

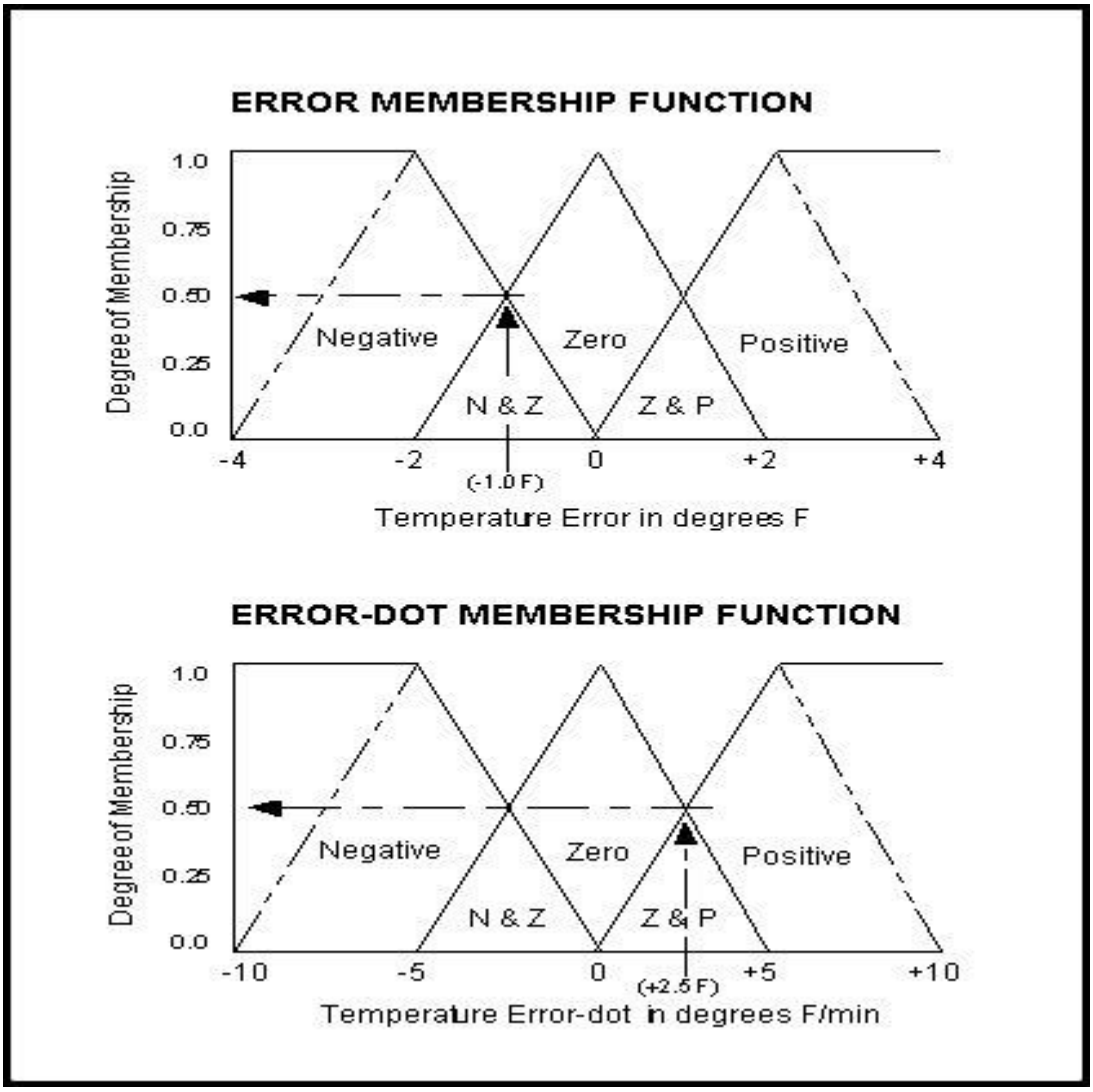


Fig 2.4: Degree of membership for the error and error-dot functions in the current example

1. If (e < 0) AND (er < 0) then Cool 0.5 & 0.0 = 0.0
2. If (e = 0) AND (er < 0) then Heat 0.5 & 0.0 = 0.0
3. If (e > 0) AND (er < 0) then Heat 0.0 & 0.0 = 0.0
4. If (e < 0) AND (er = 0) then Cool 0.5 & 0.5 = 0.5
5. If (e = 0) AND (er = 0) then No_Change 0.5 & 0.5 = 0.5
6. If (e > 0) AND (er = 0) then Heat 0.0 & 0.5 = 0.0
7. If (e < 0) AND (er > 0) then Cool 0.5 & 0.5 = 0.5
8. If (e = 0) AND (er > 0) then Cool 0.5 & 0.5 = 0.5
9. If (e > 0) AND (er > 0) then Heat 0.0 & 0.5 = 0.0

The inputs are combined logically using the AND operator to produce output response values for all expected inputs. The active conclusions are then combined into a logical sum for each membership function. A firing strength for each output membership function is computed. All that remains is to combine these logical sums in a defuzzification process to produce the crisp output.

2.7 Inferencing

The logical products for each rule must be combined or inferred (max-min'd, max-dot'd, averaged, root-sum-squared, etc.) before being passed on to the defuzzification process for crisp output generation. Several inference methods exist.

The MAX-MIN method tests the magnitudes of each rule and selects the highest one. The horizontal coordinate of the "fuzzy centroid" of the area under that function is taken as the output. This method does not combine the effects of all applicable rules but does produce a continuous output function and is easy to implement.

The MAX-DOT or MAX-PRODUCT method scales each member function to fit under its respective peak value and takes the horizontal coordinate of the "fuzzy" centroid of the composite area under the function(s) as the output. Essentially, the member function(s) are shrunk so that their peak equals the magnitude of their respective function ("negative", "zero", and "positive"). This method combines the influence of all active rules and produces a smooth, continuous output.

The AVERAGING method is another approach that works but fails to give increased weighting to more rule votes per output member function. For example, if three "negative" rules fire, but only one "zero" rule does, averaging will not reflect this difference since both averages will equal 0.5. Each function is clipped at the average and the "fuzzy" centroid of the composite area is computed.

The ROOT-SUM-SQUARE (RSS) method combines the effects of all applicable rules, scales the functions at their respective magnitudes, and computes the "fuzzy" centroid of the composite area. This method is more complicated mathematically than other methods, but was selected for this example since it seemed to give the best weighted influence to all firing rules.

2.8 Defuzzification

The RSS method was chosen to include all contributing rules since there are so few member functions associated with the inputs and outputs. For the ongoing example, an error of -1.0 and an error-dot of +2.5 selects regions of the "negative" and "zero" output membership functions. The respective output membership function strengths (range: 0-1) from the possible rules (R1-R9) are:

> Negative = (R1^2 + R4^2 + R7^2 + R8^2) (Cooling) = (0.00^2 + 0.50^2 + 0.50^2 + 0.50^2)^0.5 = 0.866
> Zero = (R5^2)^0.5 = (0.50^2)^0.5 (No Change) = 0.500
> Positive = (R2^2 + R3^2 + R6^2 + R9^2) (Heating) = (0.00^2 + 0.00^2 + 0.00^2 + 0.00^2)^0.5 = 0.000

A “FUZZY CENTROID” ALGORITHM

The defuzzification of the data into a crisp output is accomplished by combining the results of the inference process and then computing the "fuzzy centroid" of the area. The weighted strengths of each output member function are multiplied by their respective output membership function centre points and summed. Finally, this area is divided by the sum of the weighted member function strengths and the result is taken as the crisp output. One feature to note is that since the zero centre is at zero, any zero strength will automatically compute to zero. If the centre of the zero function happened to be offset from zero (which is likely in a real system where heating and cooling effects are not perfectly equal), then this factor would have an influence.

$$\frac{(neg\_center * neg\_strength + zero\_center * zero\_strength + pos\_center * pos\_strength)}{(neg\_strength + zero\_strength + pos\_strength)} = OUTPUT$$

$$\frac{(-100 * 0.866 + 0 * 0.500 + 100 * 0.000)}{(0.866 + 0.500 + 0.000)} = 63.4\%$$

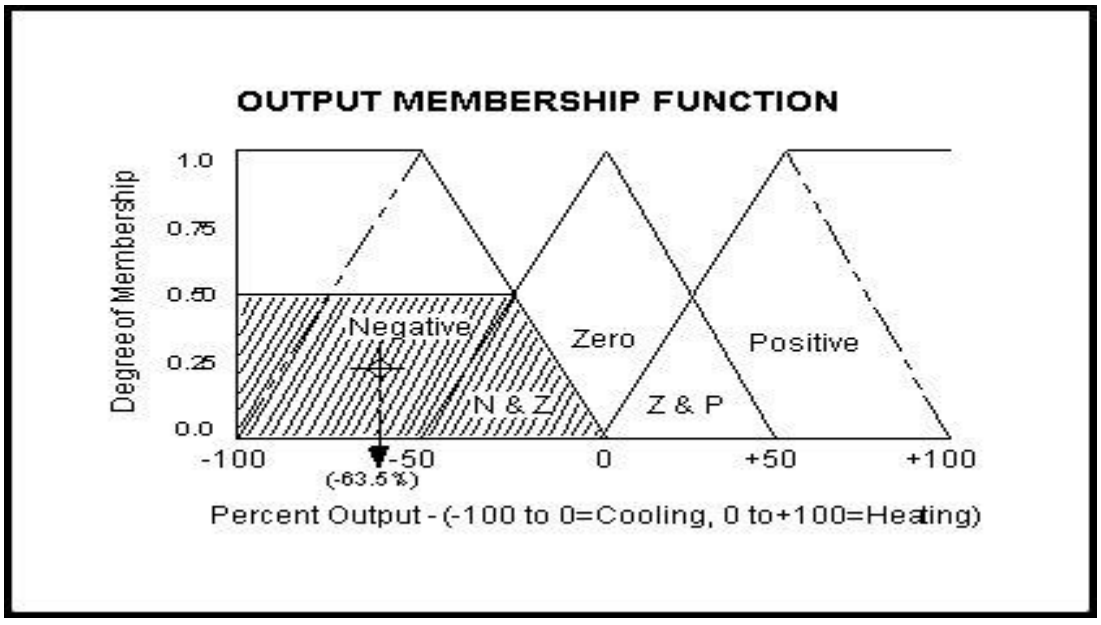


Fig 2.5: The horizontal coordinate of the centroid is taken as the crisp output.

The horizontal coordinate of the centroid of the area marked in Figure 8 is taken as the normalized, crisp output. This value of -63.4% (63.4% Cooling) seems logical since the particular input conditions (Error=-1, Error-dot=+2.5) indicate that the feedback has exceeded the command and is still increasing therefore cooling is the expected and required system response.

3. IMPLEMENTATION

The project is aimed to show whether the surrounding temperature is more or less than the predefined threshold value. Threshold value has been decided upon a reading of 60 of temperature sensor. For this purpose, two fuzzy variables that have been used are:

- Temperature read from the sensor (temp).
- Difference (diff)

diff=temp-threshold
(threshold=60)

3.1 Implementation Requirements

Input: Room Temperature

Output: LED Indicators (Red & Blue)

Rule Matrix:

Difference in Temperature	Temperature value by sensor: Low	Normal	High
Low	Red-OFF Blue-ON 1	Red-OFF Blue-ON 2	Red-ON Blue-OFF 3
Normal	Red-OFF Blue-ON 4	Red-OFF Blue-OFF 5	Red-ON Blue-OFF 6
High	Red-OFF Blue-ON 7	Red-ON Blue-OFF 8	Red-ON Blue-OFF 9

Fig 3.1: Rule Matrix for the project

Membership Functions:

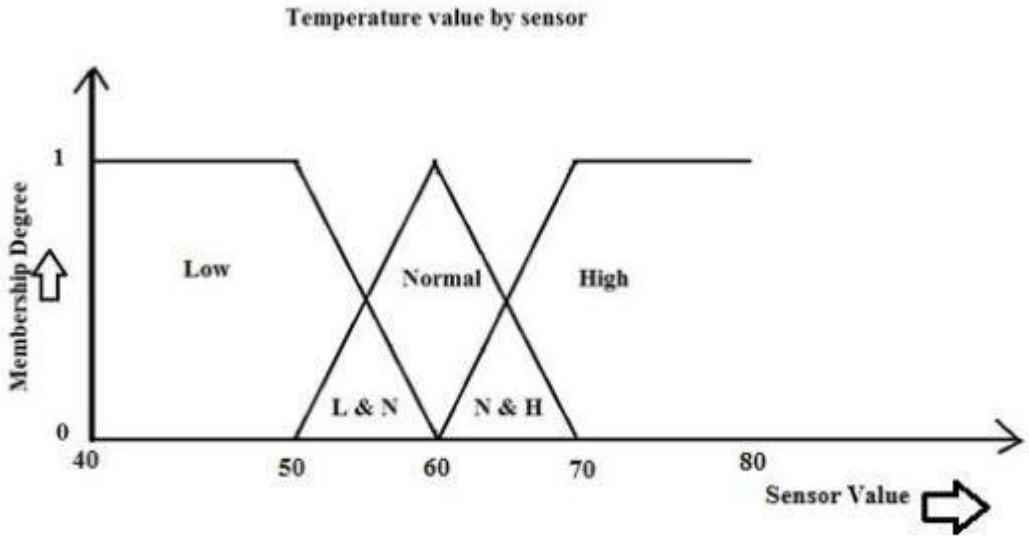


Figure 3.2 – Member function.

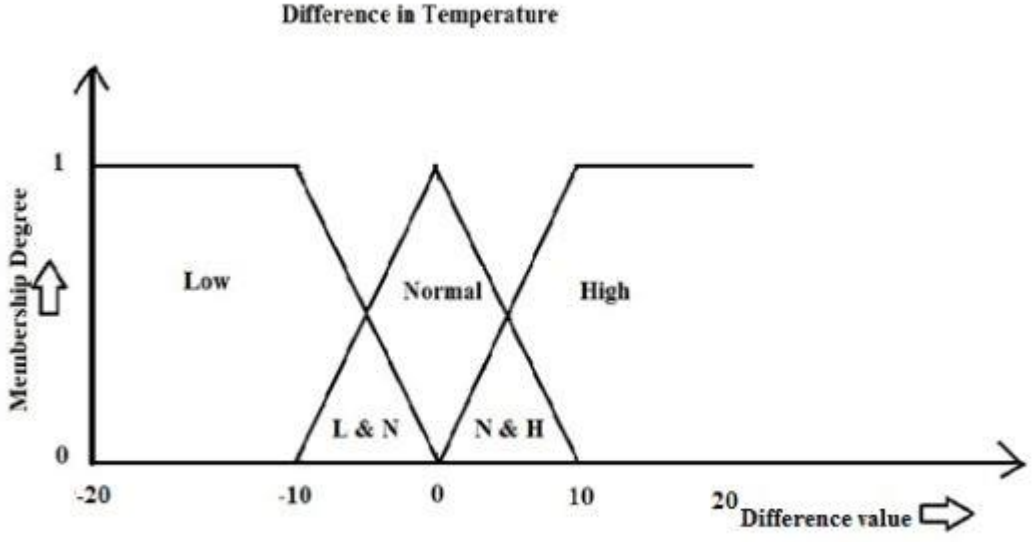


Figure 3.3 – Member function.

3.2 Source Code

```
const int analogInPin = A0;  // Analog input pin that the
LM35 is attached to

const int analogOutPin1 = 9; // Analog output pin that
the Red LED is attached to

const int analogOutPin2 = 10; // Analog output pin that
the Blue LED is attached to

double sense=1.0;        // value read from the sensor

float low, norm, high, diff;

float dlow, dnorm, dhigh;

float flow, fnorm, fhigh;

float temp[] = {0, 0, 0, 0, 0, 0, 0, 0, 0};

float difft[] = {0, 0, 0, 0, 0, 0, 0, 0, 0};

float rule[] = {0, 0, 0, 0, 0, 0, 0, 0, 0};

int out;

void setup()

{

     // initialize serial communications at 9600 bps:

     Serial.begin(9600);

}

void loop()

{

     high=0; low=0; norm=0;diff=0;

     dlow=0; dnorm=0;dhigh=0;

     flow=0; fnorm=0; fhigh=0;

     for(int i=0; i<=9;i++)

          temp[i]=difft[i]=rule[i]=0;
```

```
sense = analogRead(analogInPin);
Serial.print("sensor = " );
Serial.println(sense);
if(sense<=50)
     low=1.0;
else if(sense>50 && sense<=60)
{
     low=(-(sense)/10)+6.0;
     norm=(sense/10)-5.0;
}
else if(sense>60 && sense<=70)
{
     norm=((-(sense))/10)+7.0;
     high=(sense/10)-6.0;
}
else if(sense>70)
     high=1.0;
Serial.print("low= ");
Serial.print(low);
Serial.print(" high= ");
Serial.print(high);
Serial.print(" norm= ");
Serial.println(norm);
diff=sense-60.0; //since normal=60
Serial.print("diff= ");
Serial.println(diff);
if(diff<=-10.0)
```

```
     dlow=1;
else if(diff>-10.0 && diff<=0.0)
{
     dlow=-diff/10;
     dnorm=diff/10+1.0;
}
else if(diff>0.0 && diff<=10.0)
{
     dnorm=-diff/10+1.0;
     dhigh=diff/10;
}
else if(diff>10.0)
     dhigh=1;
Serial.print("dlow= ");
Serial.print(dlow);
Serial.print(" dhigh= ");
Serial.print(dhigh);
Serial.print(" dnorm= ");
Serial.println(dnorm);
//rule base for temp
if(low>0.0)
     temp[1]=temp[4]=temp[7]=low;
if(norm>0.0)
     temp[2]=temp[5]=temp[8]=norm;
if(high>0.0)
     temp[3]=temp[6]=temp[9]=high;
//rule base for error
```

```
     if(dlow>0.0)
          difft[1]=difft[2]=difft[3]=dlow;
     if(dnorm>0.0)
          difft[4]=difft[5]=difft[6]=dnorm;
     if(dhigh>0.0)
          difft[7]=difft[8]=difft[9]=dhigh;
     //intersection
          for(int i=0; i<=9; i++)
     {
               if(temp[i]>0.0 && difft[i]>0.0)
          {
                    rule[i]=min(temp[i], difft[i]);
               Serial.print("rule= ");
               Serial.println(i);
          }
     }
     //calculation
     flow=sqrt(sq(rule[1])+sq(rule[2])+sq(rule[4])+sq(rul
e[7]));
     fhigh=sqrt(sq(rule[3])+sq(rule[6])+sq(rule[8])+sq(ru
le[9]));
     fnorm=rule[5];
     out=(flow*(-255)  +  fnorm*0  +  fhigh*255)/(flow  +
fhigh + fnorm);
     if(out<0)
     {
          out=-(out);
          Serial.print("\t Output (cold)= ");
```

```
            Serial.println(out);
            analogWrite(analogOutPin1, 0);
            analogWrite(analogOutPin2, out);
      }
      else if(out>0)
      {
            Serial.print("\t Output (hot)= ");
            Serial.println(out);
            analogWrite(analogOutPin2, 0);
            analogWrite(analogOutPin1, out);
      }
      else
      {
            Serial.print("\t Output (No Change)= ");
            Serial.println(out);
            analogWrite(analogOutPin2, 0);
            analogWrite(analogOutPin2, 0);
      }
      // wait 200 milliseconds before the next loop
      // for the analog-to-digital converter to settle
      // after the last reading:
      delay(200);
}
```

This is the source code which was used for the implementation of temperature control system. For this, two LED's are used to show the output. Red is used to indicate that the system is hot and the Blue is used to indicate that the system is cold.

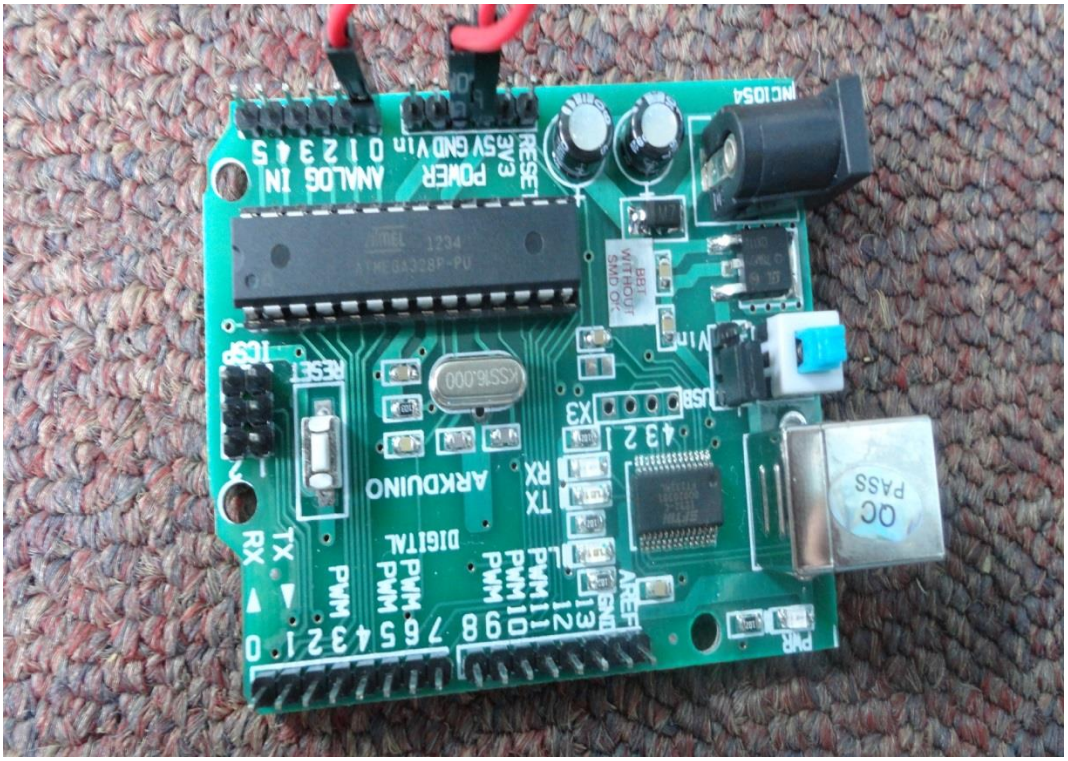
3.3 Components Used

3.3.1 Hardware Components

A. Arduino Microcontroller Board

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing).

This is the arduino board which we have used:

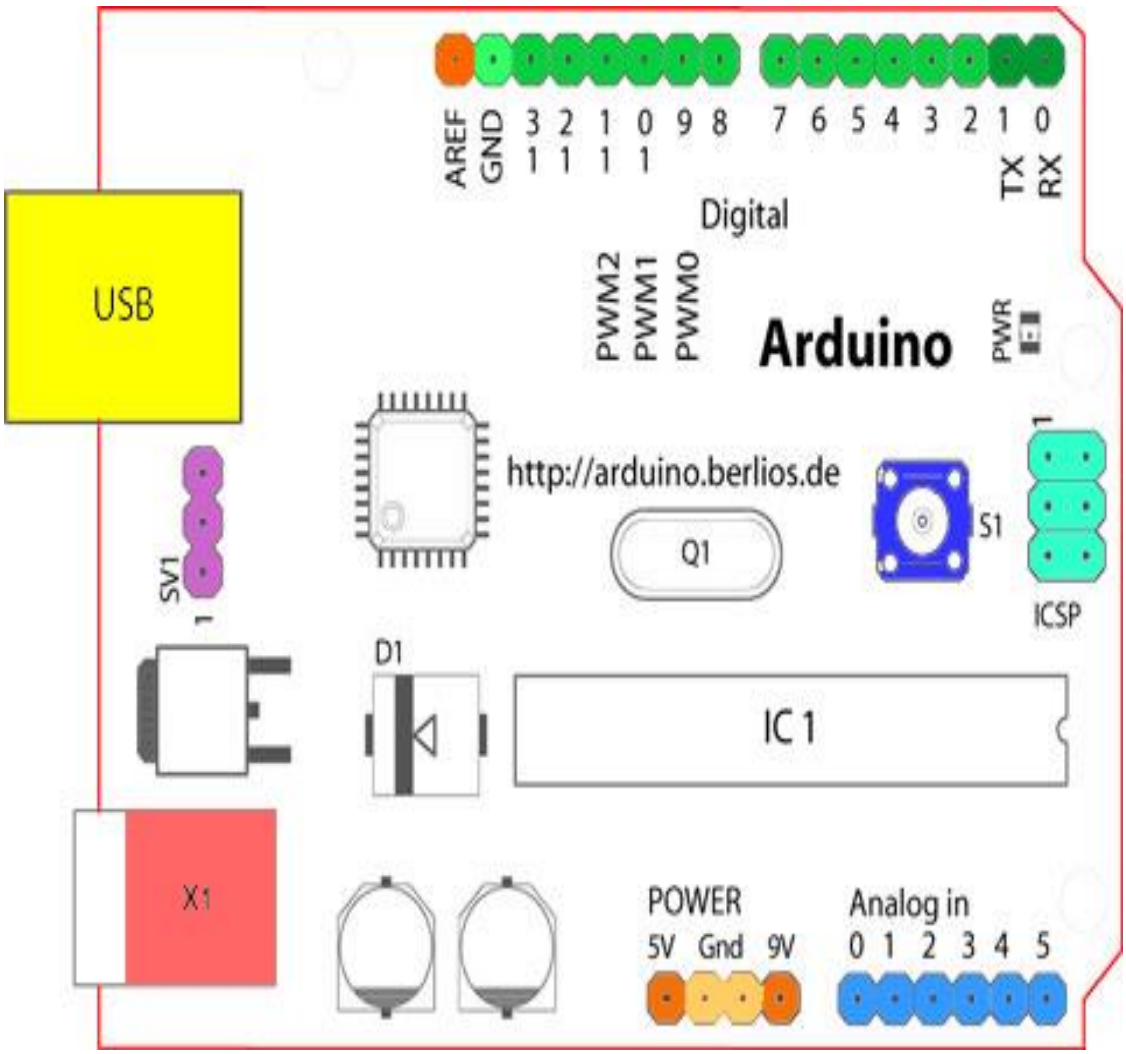


The board is known as Arduino Duemilanove w/ ATmega328. It is connected to the USB port of the computer.

It has following pins:

- Six Analog in pins named from 0-5
- One analog reference pin
- One digital ground
- Two ground pins

- Twelve digital pins named from 2-13
- Digital Pins 0-1/Serial In/Out - TX/RX (These pins are not used for digital I/O)
- Reset Button
- In-circuit Serial Programmer named as ICSP as shown in the figure.
- External power supply-in
- USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board)
- Integrated Circuit
- Power pin

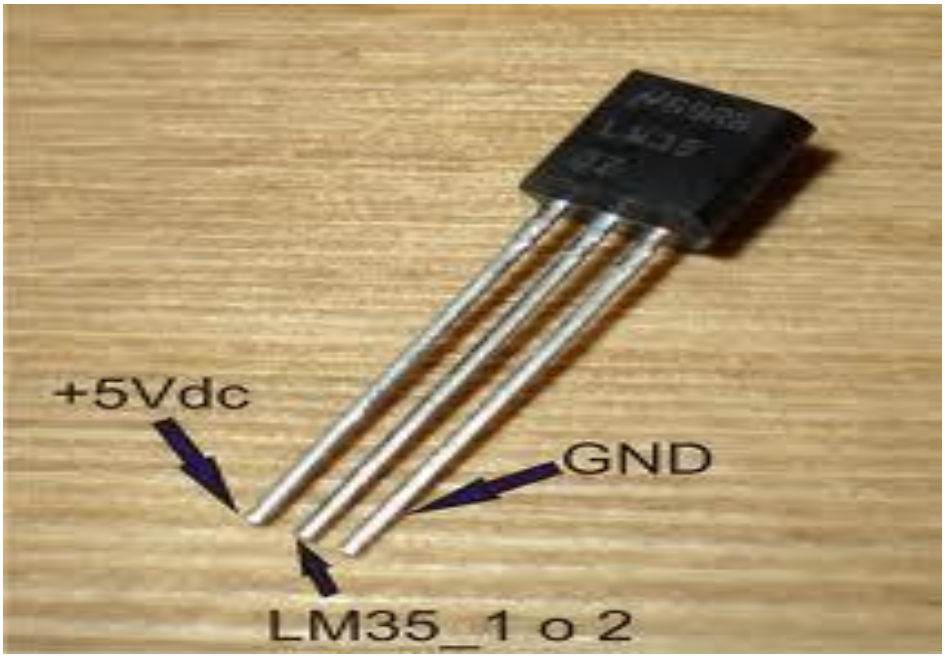


B. LM35 – Temperature Sensor

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ˚ Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling.

Features:

- Calibrated directly in ˚Celsius (Centigrade)
- Linear + 10.0 mV/˚C scale factor
- Rated for full −55˚ to +150˚C range
- Operates from 4 to 30 volts



.

The LM35 does not require any external calibration or trimming to provide typical accuracies of ± ¼ degree Celsius at room temperature and ± ¾ degree Celsius over a full -55 to +150 degree Celsius range.

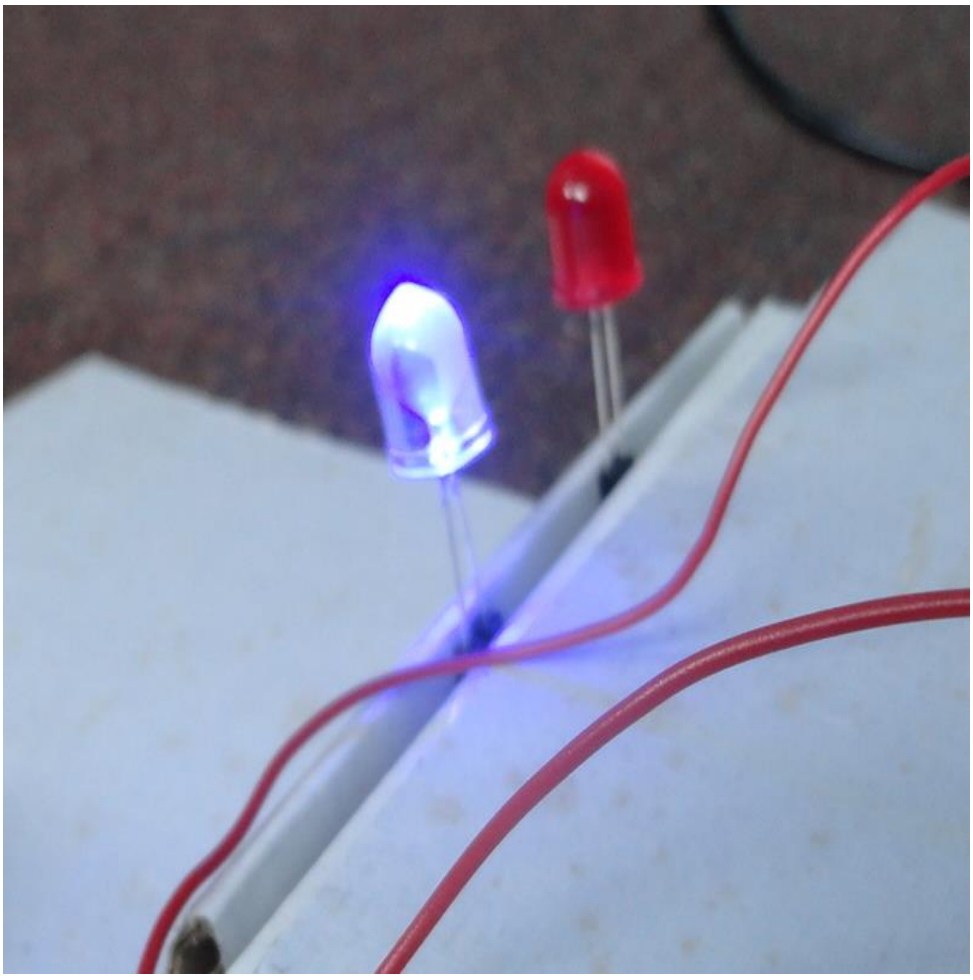
C. LED’s

Two LED’s are used for this purpose.

Red for showing that the temperature sensed by the sensor is hot and blue to show that the temperature sensed by the sensor is cold.

Whenever, for example, sensor is put closed to a bottle filled with ice, blue one will glow.

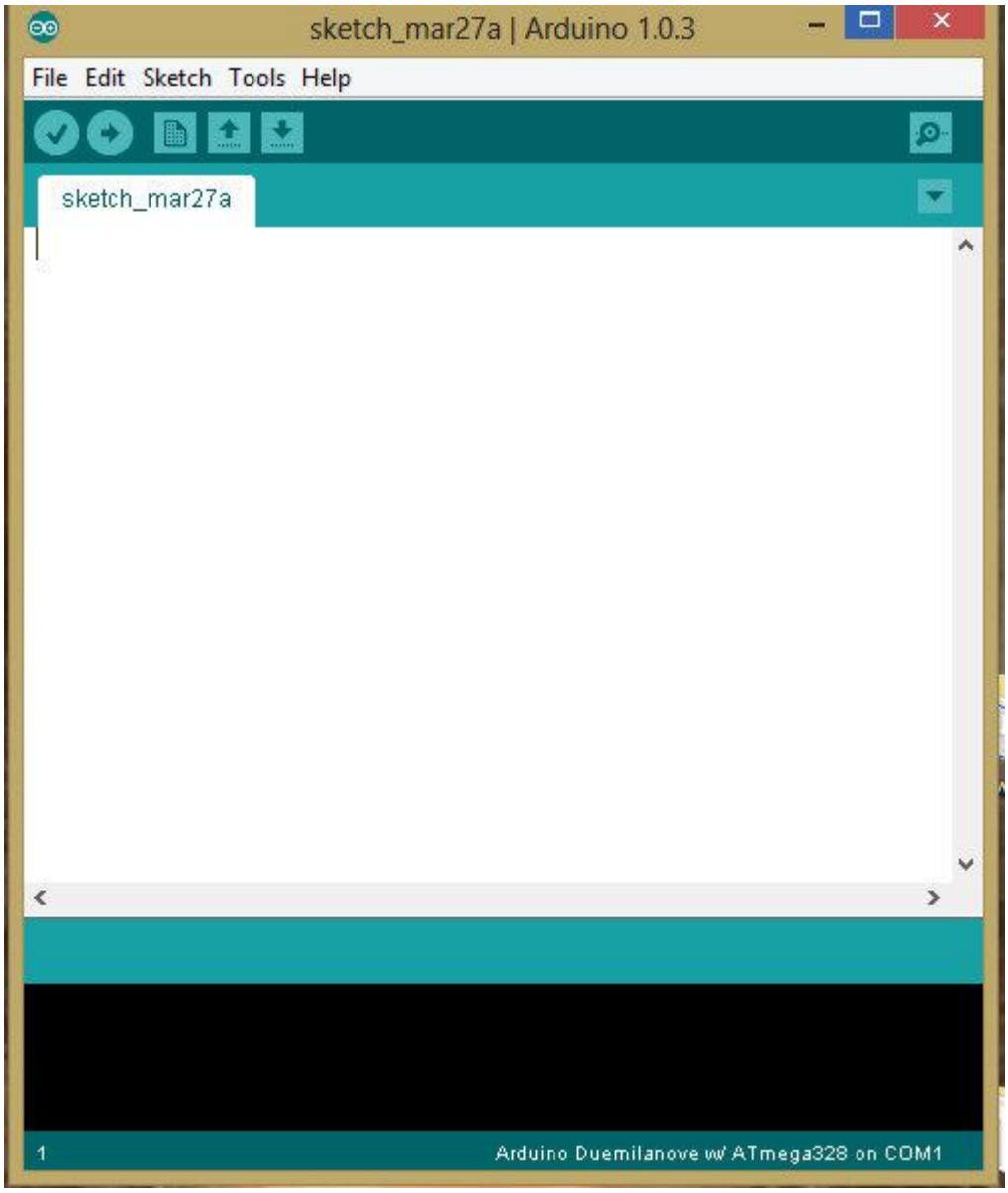
Red one will glow whenever the sensor is put close to a bottle filled with hot water or a candle.



3.3.2 Software Components

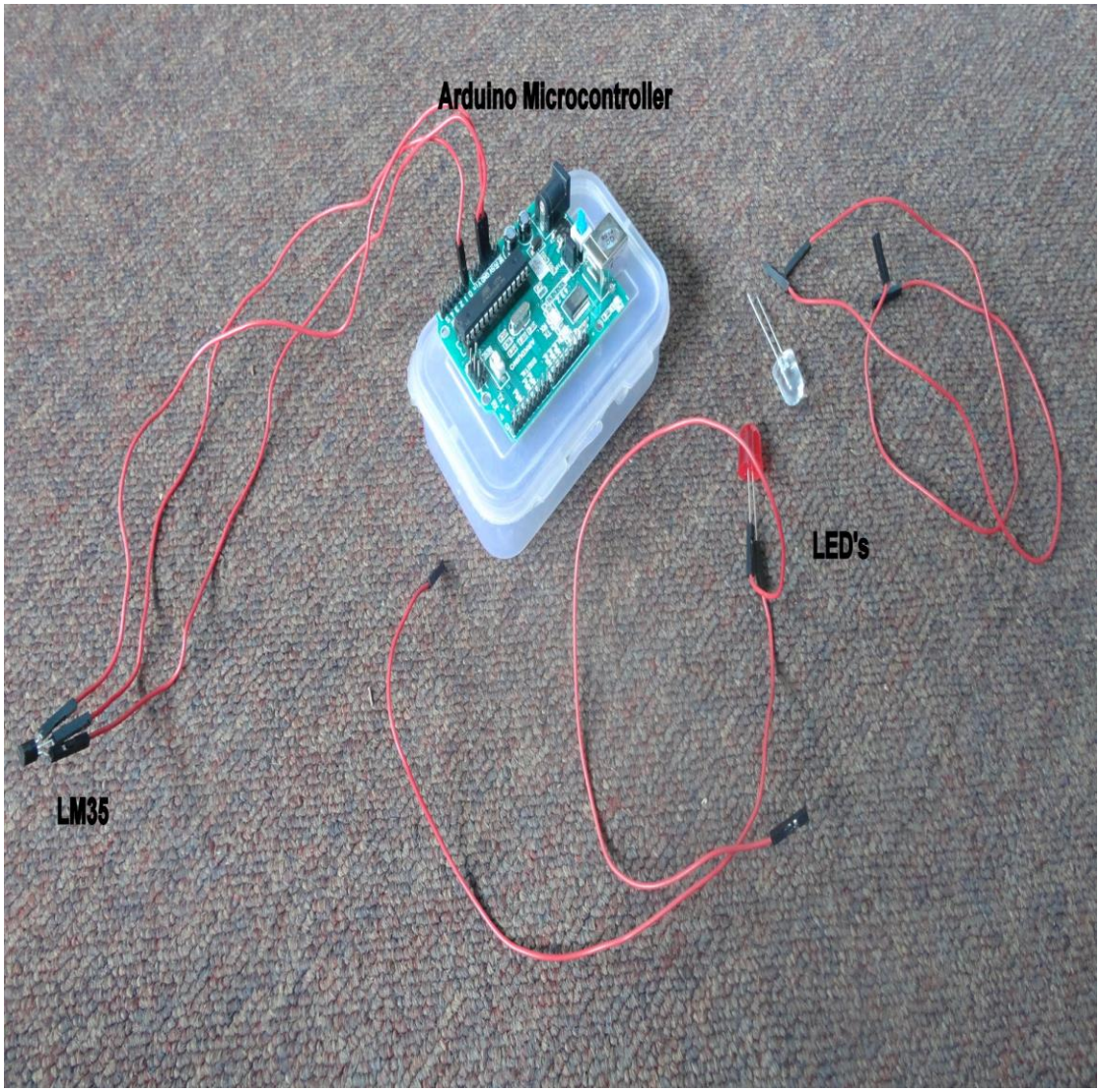
Arduino Development Tool:

The open-source Arduino environment makes it easy to write code and upload it to the input/output board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java. The code is then verified on the compiler. If it gets compiled successfully then it shows that the code has no errors.



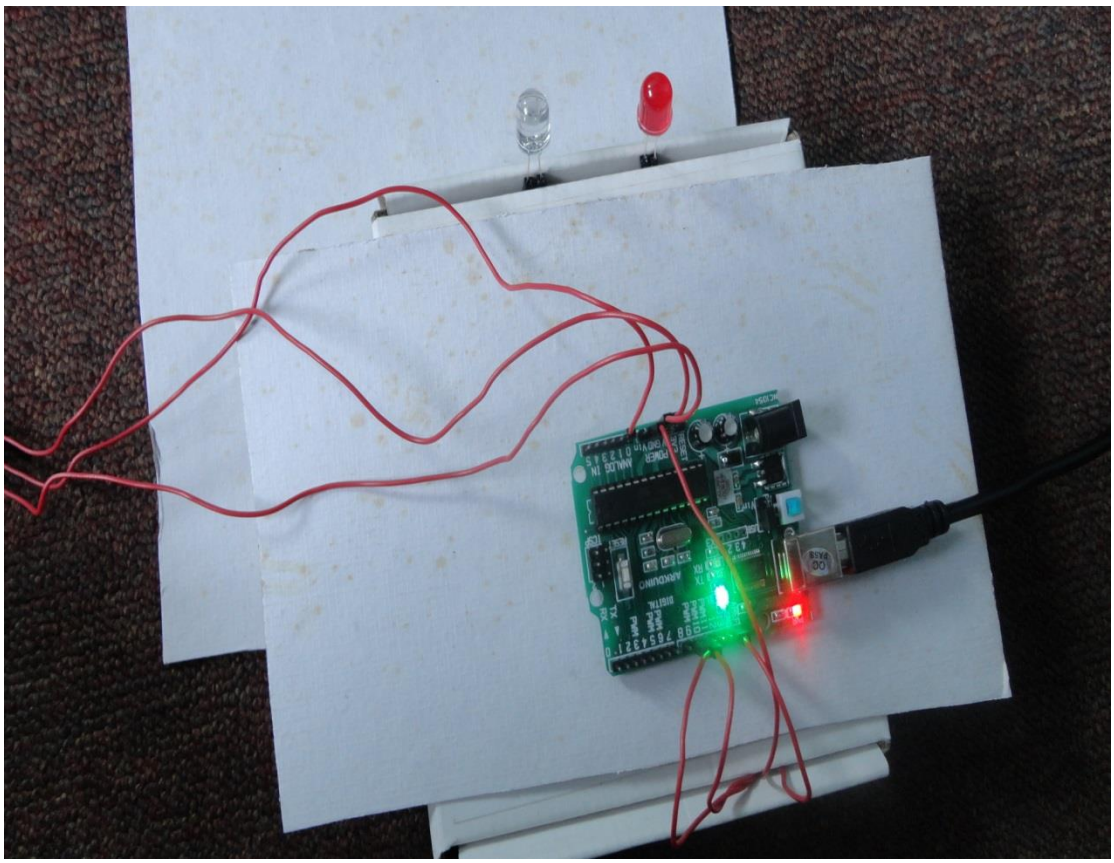
3.4 Pictures

This is the circuit of the temperature control system which we have implemented. It contains arduino microcontroller, two LED's and LM35 temperature sensor. Arduino board has a power supply that can be directly given and USB which is connected to the computer. One pin of LM35 is connected to the analog ground of the arduino board, second pin is connected to the analog in pin A0 and the third pin is connected to the voltage pin of the arduino board.



This picture shows when the board is connected to the computer. Initially the blinking code triggers off and hence green light blinks continuously. Red light shows that board is connected to the computer and is on com3.

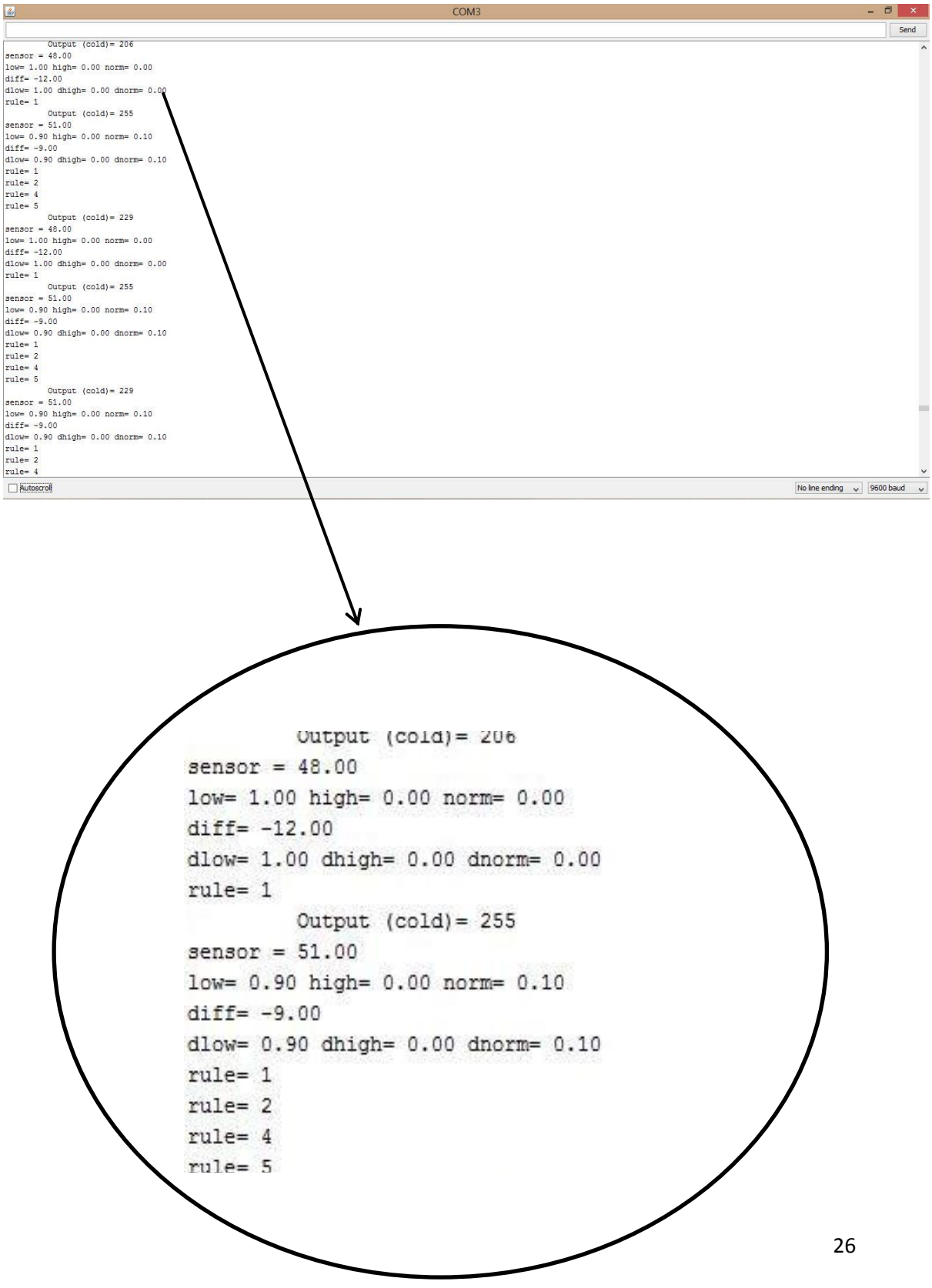
The power can be directly given to the arduino board when connected to the computer or else it can be given by the supply.



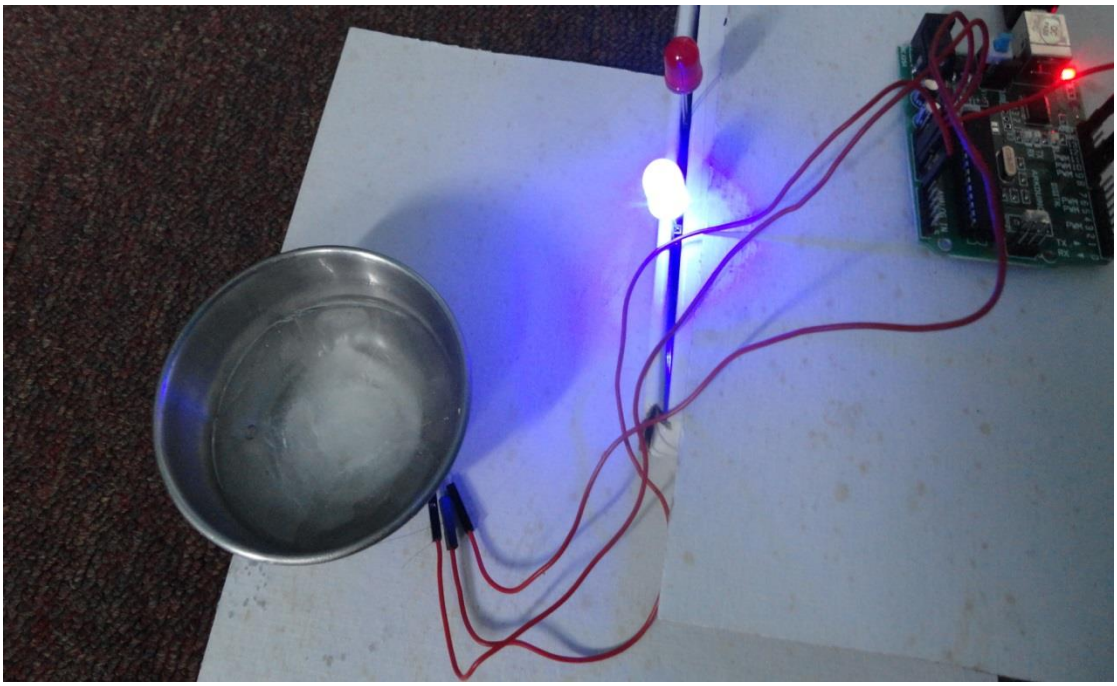
4. RESULT

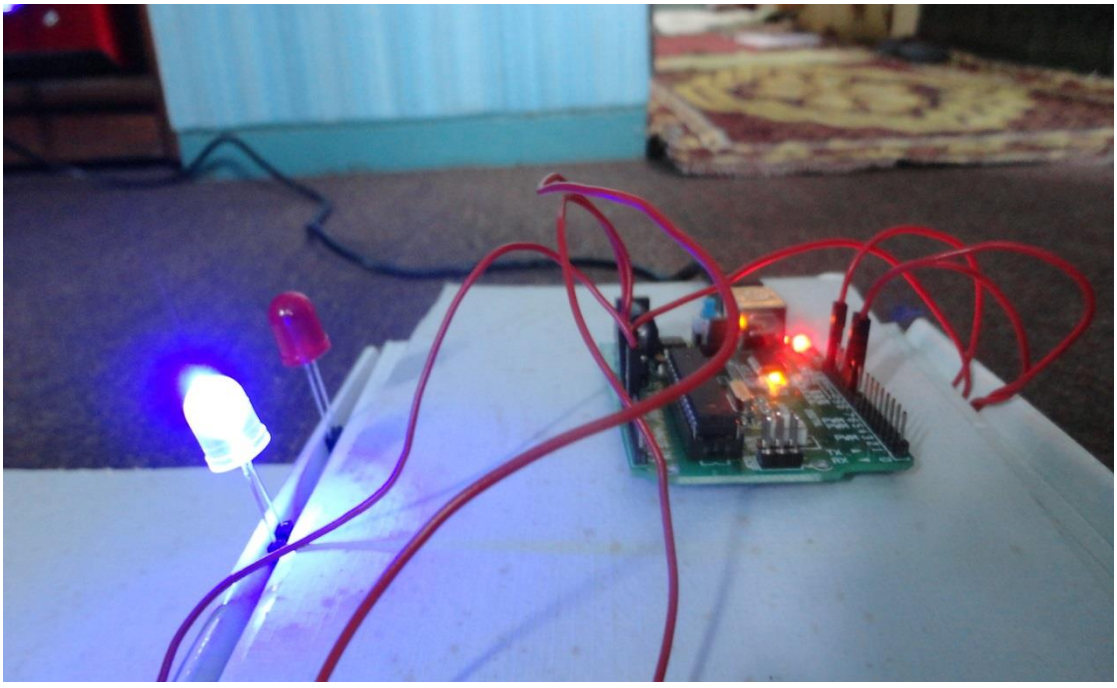
4.1 Result Screenshots

This is the screenshot of the serial monitor which shows the output as cold when the sensor is put near ice.

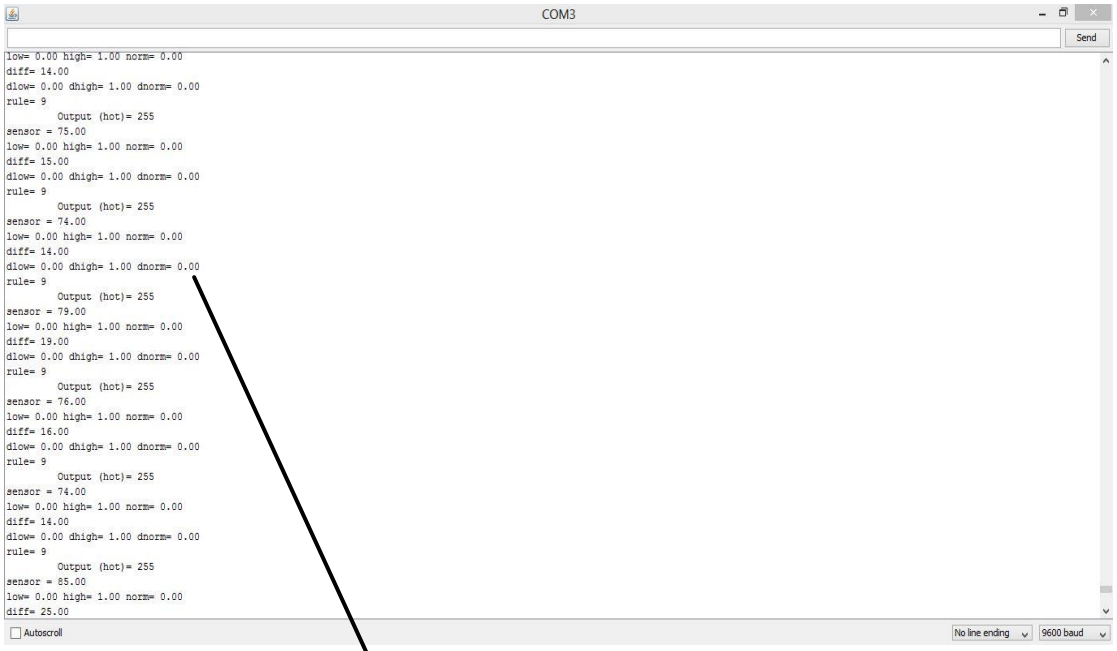


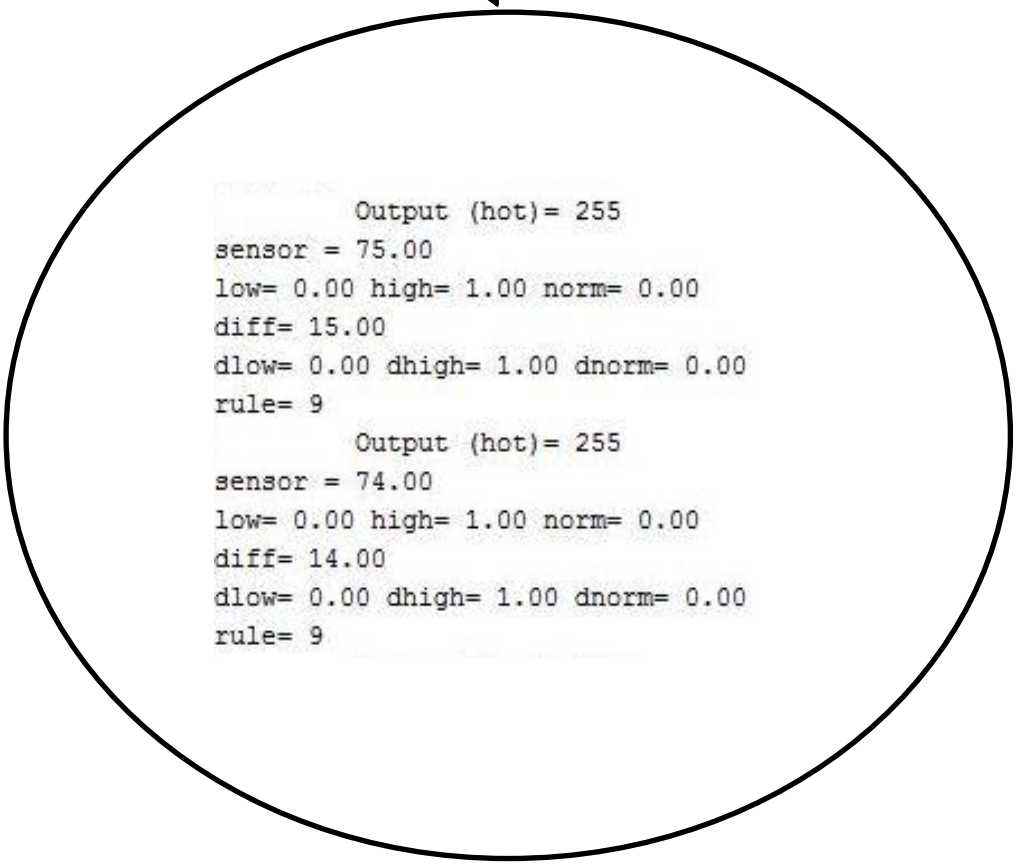
This is the screenshot of the whole system with the blue LED glowing which depicts that the system close to the sensor is cold. For that time, red LED is off.



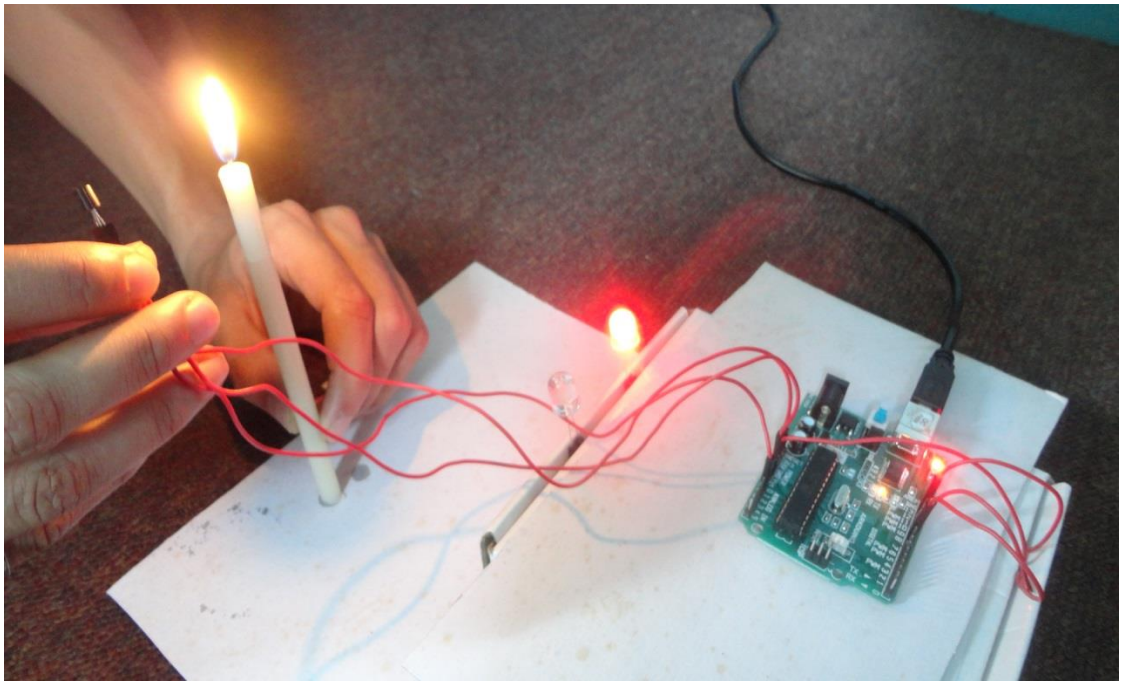


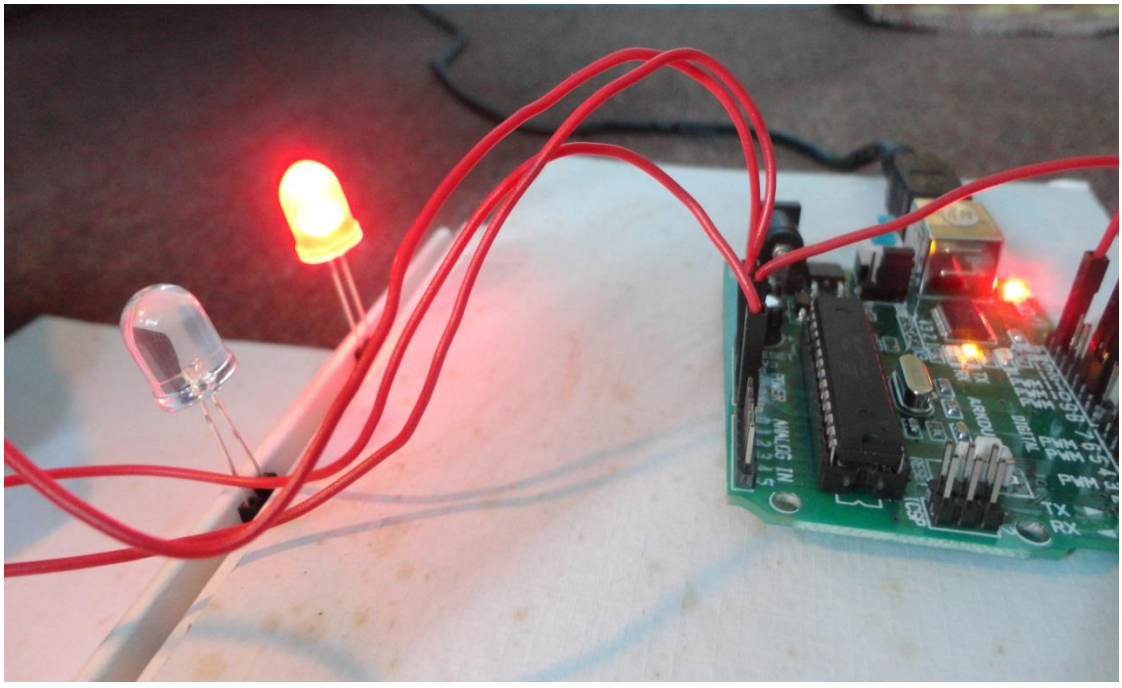
This is the screenshot of the serial monitor which shows the output as hot when the sensor is put near candle.





This is the screenshot of the whole system with the red LED glowing which depicts that the system close to the sensor is hot. For that time, blue LED is off.





4.2 Applications

Temperature Control System has many applications on a large scale.

Air Conditioners:

The task of dehumidification and temperature decrease goes hand in hand in case of conventional AC. Once target temperature is reached AC seizes to function like a dehumidifier. Also complex interactions between user preferences, actual room temperature and humidity level are very difficult to model mathematically. The optimal limits of comfort zone, typically marked at a temperature of 25°C and dew point 11°C, are used as the targets. For instance at higher humidity level (say at dew point 18°C) an occupant may perceive same comfort level at 22ºC as he would perceive at 26ºC at dew point 15°C. This translates to huge energy and monitory saving in terms of reduced compressor/fan duty cycle. In the developed scheme, the sensor captured temperature, user temperature preference and humidity readings are fuzzified.

Preterm Neonate Incubators:

This employs the use of Fuzzy Logic Control (FLC) Methodology to control the supply of heat into the Neonate Incubator in order to attain thermoneutrality. Fuzzy Logic Controllers have proven to be as efficient as Conventional Controllers and sometimes the only alternatives suitable for designing controllers for complex processes/plants or for processes that are non-linear in nature. In this research, a 2 inputs,2 outputs, 18 MISO Rules PD Fuzzy Logic Temperature Controller is designed and implemented. This research work did not only show the applicability of Fuzzy Logic Control Methodology to controlling the Incubator Temperature to attain thermoneutrality but also the efficient stabilization of the Incubator Temperature at a desired value and thereby prevention of hypothermia/hyperthermia related diseases/conditions and death. Also it eliminates the error-prone human control of the heating element of the Neonate Incubator. The performance evaluation of the designed Fuzzy Logic Controller was conducted through the examination of the stability, rise time, over/under shoot of the controller in a Computer Simulation using C++ programming Language.

4.3 Conclusion:

The report explains what fuzzy logic is and how it works by using the example of a Temperature Control System using Fuzzy Logic.

The project implementation involves the use of a Arduino board, a microcontroller which has been used to implement a Temperature Control System which uses Fuzzy Logic. The microcontroller takes two inputs, the surrounding temperature, using an LM35 temperature sensor, and the difference between this value and the ambient temperature value of about 30 degrees Celsius. The microcontroller performs fuzzification, inferenceing, and defuzzification to obtain an output value. The microcontroller displays this value by glowing of one LED. The blue LED glows for lower temperatures and red glows for higher temperatures.

According to Lotfi Zadeh, fuzzy logic with its support for various shades of grey is a far better and more realistic representation of reality and the world we live in. But that doesn't mean that there is any ambiguity in fuzzy logic itself, as the name "fuzzy" may otherwise suggest.

As Dr. Zadeh says, fuzzy logic "is a precise logic of imprecision".

4.4 Further Work:

As an improvement to this system, it is possible to take the required temperature from the user via a 3 by 4 numerical keypad or even a remote. Also, a heating system can be added to the system to provide a higher temperature whenever required.

REFERENCES

[1] "Fuzzy Sets and Applications: Selected Papers by L.A. Zadeh", ed. R.R. Yager et al. (John Wiley, New York, 1987).

[2] "Fuzzy Fundamentals" by E. Cox (IEEE Spectrum, October 1992, pp. 58-61).

[3] "Fundamentals of Fuzzy Logic: Parts 1,2,3" by G. Anderson (SENSORS, March-May 1993).

[4] "Industrial Applications of Fuzzy Control" ed. M. Sugeno (North-Holland, New York, 1985).

[5] "Fuzzy Logic - From Concept to Implementation", (Application Note EDU01V10-0193. ((c) 1993 by Aptronix, Inc, (408) 428-1888).

[6] "Fuzzy Motor Controller" Huntington Technical Brief, D. Brubaker ed. (April 1992, No. 25, Menlo Park, CA 1992).

ACKNOWLEDGEMENT

We thank Ms. Ruhi K. Bajaj for her guidance and assistance in the success of this project. This project would not have been implemented successfully without her constant support. We also thank the Mumbai University for giving this opportunity to understand Fuzzy Logic and its implementation in various systems, specifically Temperature Control System.