CHAPTER 1
INTRODUCTION

A temperature controller is a device that is used to control temperature. It does this by first measuring the temperature (process variable), it then compares it to the desired value (set value). The difference between these values is known as the error (Deviation). Temperature controllers use this error to decide how much heating or cooling is required to bring the process temperature back to the desired value. This project uses fuzzy controller to obtain the control signal that is used to trigger the SCR that controls the heating device. [1]

Fuzzy logic is a mathematical system that analyzes analog input values in terms of fuzzy variables that takes continuous values between 0 and 1, in contrast to classical or digital logic, which operates on discrete values of either 0 or 1. A fuzzy logic based design control system offers flexibility in system design and implementation, since its implementation uses “if then” logic instead of sophisticated differential equations. The fuzzy controller is developed using MATLAB. Here we used Fuzzy Logic Toolbox which is very useful software for development and testing of Fuzzy Logic system. It can be very quickly implemented and its visual impact is very encouraging.

1.1 PROJECT OUTLINE

Our proposed system contains two main parts: the power circuit and the control circuit.The modification of the temperature is controlled by means of a heating resistor whose AC heating power is modulated by a phase-control SCR. The SCR is driven by a specialized TCA785 circuit located on the phase control board.The necessary voltage adaptation between the DAC and the phase control circuit is assured through a voltage amplifier based on a rail-to-rail AD820 operational amplifier.The Arduino Uno board is the “brain” of the entire system. It is primarily responsible for the update of the digital control signal at every time instance.

CHAPTER 2
PROJECT DETAILS

2.1 HARDWARE SETUP

The device whose temperature is under obsrvation is a 2Ω heating resistor. The aim is to maintain a constant temperature using the control mechanism. A FLC is designed in MATLAB whose control output is given to the DAC(MCP4725). The converted analog signal after amplification is used to trigger the SCR via the phase control board(TCA785).

2.1.1 ARDUINO UNO BOARD

Arduino Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. [2]

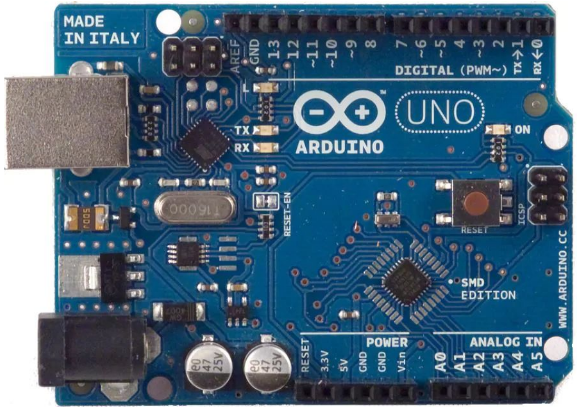


Fig. 2.1: Arduino Uno Board

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

Table No. 2.1: Arduino specification

The Arduino UNO can be powered via the USB connection or with an external power supply. The power source is selected automatically. The board can operate on an external supply of 6 to 20 volts. Each of the 14 digital pins on the UNO can be used as an input or output,using pinMode() , digitalWrite() , and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor.The Arduino UNO has several facilities for communicating with a computer, another Arduino,or other microcontrollers. The ATmega328 provides UART, TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX).

The Arduino UNO can be programmed with the Arduino software. The ATmega328 on the Arduino UNO comes preburned with a bootloader that allows to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol. The Arduino UNO has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Input and output

• Serial: 0(RX) and 1(TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip

•External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

•PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite()
function.

• SPI:10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

• LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

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

• I2C: 4 (SDA) and 5 (SCL). Support I2C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

• AREF:- Reference voltage for the analog inputs. Used with analogReference().

• Reset:-Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

2.1.2 TEMPERATURE SENSOR- DS18B20

The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply. [3]

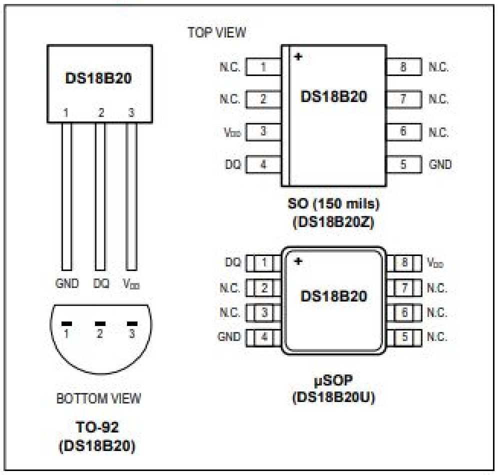
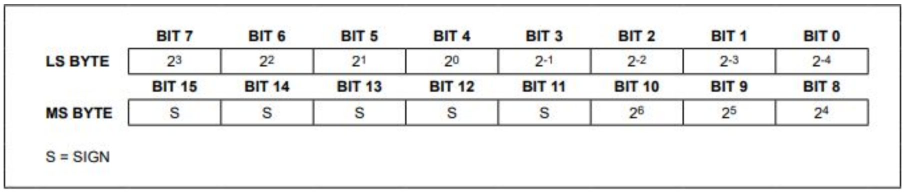


Fig. 2.2: Pin configuration of DS18B20

The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the temperature sensor is user-configurable to 9, 10, 11, or 12 bits, corresponding to increments of 0.5°C, 0.25°C, 0.125°C, and 0.0625°C, respectively. The default resolution at power-up is 12-bit. The sign bits (S) indicate if the temperature is positive or negative: for positive numbers S = 0 and for negative numbers S = 1. If the DS18B20 is configured for 12-bit resolution, all the bits in the temperature register will contain valid data. For 11-bit resolution, bit 0 is undefined. For 10-bit resolution, bits 1 and 0 are undefined, and for 9-bit resolution bits 2, 1, and 0 are undefined.



	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
LS BYTE	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
MS BYTE	S	S	S	S	S	2^6	2^5	2^4

S = SIGN

Fig. 2.3: Temperature Register Format

Electrical Specifications

1. Supply voltage (V_{DD}): 3V - 5.5V
2. Input Logic Low(V_{IL}): -0.3V - 0.8V
3. Input Logic High(V_{IH}): 2.2V - 5.5V
4. Active Current(I_{DD}): 1mA- 1.5mA, for V_{DD} = 5V

2.1.3 PHASE CONTROL IC- TCA785

This phase control IC is intended to control thyristors, triacs, and transistors. The trigger pulses can be shifted within a phase angle between 0˚ and 180˚. Typical applications include converter circuits, AC controllers and three-phase current controllers. [4]

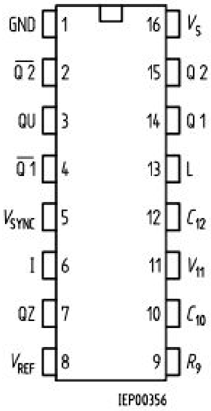


Fig. 2.4: Pin diagram of TCA785

PIN	NAME	FUNCTION
1	GND	Ground
2	$\overline{Q2}$	Output 2 inverted
3	$Q\ U$	Output U
4	$\overline{Q1}$	Output 1 inverted
5	V sync	Synchronous voltage
6	I	Inhibit
7	Q Z	Output Z
8	Vref	Reference voltage
9	R9	Ramp resistance
10	C10	Ramp Capacitance
11	V11	Control voltage
12	C12	Pulse extension
13	L	Long pulse
14	$Q1$	Output 1
15	$Q2$	Output 2
16	Vs	Supply voltage

Table No. 2.2: Pin definitions and functions of TCA785

The synchronization signal is obtained via a high-ohmic resistance from the line voltage (voltage Vs). A zero voltage detector evaluates the zero passages and transfers them to the synchronization register. This synchronization register controls a ramp generator, the capacitor C10 is charged by a constant current (determined by R9). If the ramp voltage V10 exceeds the control voltage V11 (triggering angle ϕ), a signal is processed to the logic. Dependent on the magnitude of the control voltage V11, the triggering angle ϕ can be shifted within a phase angle of 0° to 180°

Electrical Specifications

5. Supply voltage (V_s): 8V - 18V
6. Operating frequency(f): 10Hz - 500 Hz; Typical value- 50 Hz
7. Ambient Temperature(T_A): -25℃ - 85℃
8. Output Voltage: V_s
9. Output Current: 10mA

2.1.4 12-BIT DIGITAL TO ANALOG CONVERTER- MCP4725

The MCP4725 is a low-power, high accuracy, single channel, 12-bit buffered voltage output Digital-toAnalog Convertor (DAC) with non-volatile memory (EEPROM). Its on-board precision output amplifier allows it to achieve rail-to-rail analog output swing. The DAC input and configuration data can be programmed to the non-volatile memory (EEPROM) by the user using I2C interface command. [5]

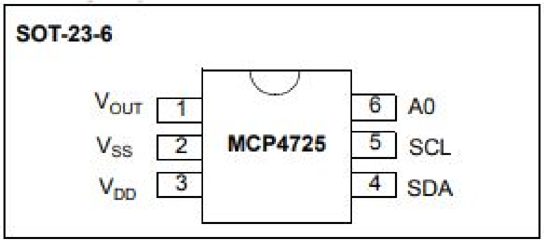


Fig. 2.5: Pin diagram of MCP4725

PIN	NAME	FUNCTION
1	V_{out}	Analog output voltage
2	V_{ss}	Ground reference
3	V_{dd}	Supply voltage
4	SDA	I^2C Serial data
5	SCL	I^2C Serial clock input
6	A0	I^2C Address bit selection pin

Table No. 2.3: Pin definitions and functions of MCP4725

The MCP4725 has an external A0 address bit selection pin. This A0 pin can be tied to VDD or VSS of the user's application board.

Electrical Specifications

1. Operating voltage (V_{dd}): 2.7V - 5.5V
2. Supply current (I_{dd}): max. 400µA
3. Power down current (I_{DDP}): max. 2µA

Chapter 3
FUZZY LOGIC

3.1 INTRODUCTION

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.

The idea of fuzzy logic was first advanced by Dr. Lotfi Zadeh of the University of California at Berkeley in the 1960s. Dr. Zadeh was working on the problem of computer understanding of natural language. Natural language (like most other activities in life and indeed the universe) is not easily translated into the absolute terms of 0 and 1. (Whether everything is ultimately describable in binary terms is a philosophical question worth pursuing, but in practice much data we might want to feed a computer is in some state in between and so, frequently, are the results of computing.) It may help to see fuzzy logic as the way reasoning really works and binary or Boolean logic is simply a special case of it. [6]

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 "tall" or "short" but ".38 of tallness."

Fuzzy logic seems closer to the way our brains work. We aggregate data and form a number of partial truths which we aggregate further into higher truths which in turn, when certain thresholds are exceeded, cause certain further results such as motor reaction. A similar kind of process is used in neural networks, expert systems and other artificial intelligence applications. Fuzzy logic is essential to the development of human-like capabilities for AI, sometimes referred to as artificial general intelligence: the representation of generalized human cognitive abilities in software so that, faced with an unfamiliar task, the AI system could find a solution.

3.2 TYPES OF FUZZY INFERENCE SYSTEMS

There are mainly two types of fuzzy inference systems which are discussed below.[7]

3.2.1 Mamdani Fuzzy Inference System

This system was proposed in 1975 by Ebhasim Mamdani. Basically, it was anticipated to control a steam engine and boiler combination by synthesizing a set of fuzzy rules obtained from people working on the system.

Steps for Computing the Output

Following steps need to be followed to compute the output from this FIS −

- Step 1 − Set of fuzzy rules need to be determined in this step.
- Step 2 − In this step, by using input membership function, the input would be made fuzzy.
- Step 3 − Now establish the rule strength by combining the fuzzified inputs according to fuzzy rules.
- Step 4 − In this step, determine the consequent of rule by combining the rule strength and the output membership function.
- Step 5 − For getting output distribution combine all the consequents.
- Step 6 − Finally, a defuzzified output distribution is obtained.

Following is a block diagram of Mamdani Fuzzy Inference System.

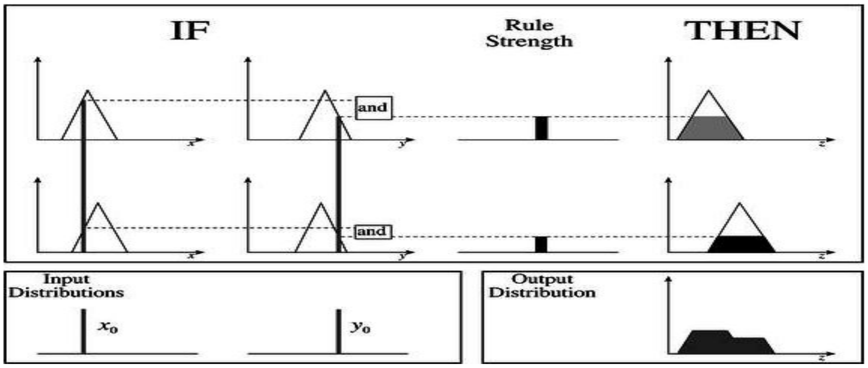


Fig. 3.1: Mamdani Fuzzy Inference System.

3.2.2 Takagi-Sugeno Fuzzy Model (TS Method)

This model was proposed by Takagi, Sugeno and Kang in 1985. Format of this rule is given as −

IF x is A and y is B THEN Z = f(x,y)

Here, *AB* are fuzzy sets in antecedents and *z = f(x,y)* is a crisp function in the consequent.

Fuzzy Inference Process

The fuzzy inference process under Takagi-Sugeno Fuzzy Model (TS Method) works in the following way −

- Step 1: Fuzzifying the inputs − Here, the inputs of the system are made fuzzy.
- Step 2: Applying the fuzzy operator − In this step, the fuzzy operators must be applied to get the output.

Rule Format of the Sugeno Form

The rule format of Sugeno form is given by −

if 7 = x and 9 = y then output is z = ax+by+c

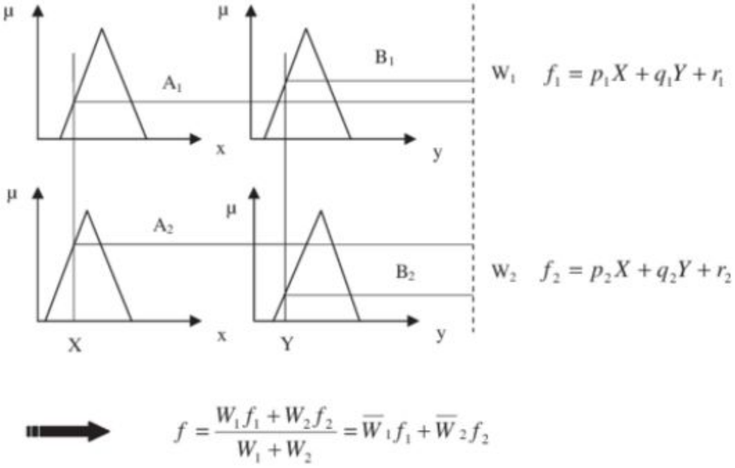

μ
μ
A1
B1
x
y
W1
$f_1 = p_1X + q_1Y + r_1$
μ
μ
A2
B2
W2
$f_2 = p_2X + q_2Y + r_2$
X
x
Y
y
$f = \frac{W_1 f_1 + W_2 f_2}{W_1 + W_2} = \overline{W}_1 f_1 + \overline{W}_2 f_2$

Fig. 3.2 : Takagi Sugeno Fuzzy Inference System

3.2.3 Comparison between the two methods

Let us now understand the comparison between the Mamdani System and the Sugeno Model.

- Output Membership Function − The main difference between them is on the basis of output membership function. The Sugeno output membership functions are either linear or constant.
- Aggregation and Defuzzification Procedure − The difference between them also lies in the consequence of fuzzy rules and due to the same their aggregation and defuzzification procedure also differs.
- Mathematical Rules − More mathematical rules exist for the Sugeno rule than the Mamdani rule.
- Adjustable Parameters − The Sugeno controller has more adjustable parameters than the Mamdani controller.

3.3 APPLICATIONS

- It is used in the aerospace field for altitude control of spacecraft and satellite.
- It has been used in the automotive system for speed control, traffic control.
- It is used for decision making support systems and personal evaluation in the large company business.
- It has applications in chemical industry for controlling the pH, drying, chemical distillation process.
- Fuzzy logic are used in Natural language processing and various intensive applications in Artificial Intelligence.
- Fuzzy logic are extensively used in modern control systems such as expert systems.
- Fuzzy Logic is used with Neural Networks as it mimics how a person would make decisions, only much faster. It is done by Aggregation of data and changing into more meaningful data by forming partial truths as Fuzzy sets.

3.4 ADVANTAGES OF FUZZY LOGIC SYSTEM

- This system can work with any type of inputs whether it is imprecise, distorted or noisy input information.
- The construction of Fuzzy Logic Systems is easy and understandable.
- Fuzzy logic comes with mathematical concepts of set theory and the reasoning for that is quite simple.
- It provides a very efficient solution to complex problems in all fields of life as it resembles human reasoning and decision making.
- The algorithms can be described with little data, so little memory is required.

3.5 DISADVANTAGES OF FUZZY LOGIC SYSTEM

- Many researchers proposed different ways to solve a given problem through fuzzy logic which lead to ambiguity.There is no systematic approach to solve a given problem through fuzzy logic.
- Proof of its characteristics is difficult or impossible in most cases because every time we do not get mathematical description of our approach.
- As fuzzy logic works on precise as well as imprecise data so most of the time accuracy is compromised.

Chapter 4
PROJECT IMPLEMENTATION

4.1 SYSTEM OVERVIEW

The process consists in maintaining a user specified constant temperature (T_{ref}) in a thermal enclosure. The current temperature, T is measured using a smart sensor DS18B20 which provides a 9 to 12-bit temperature reading over a 1-Wire interface. The modification of the temperature is controlled by means of a heating resistor whose AC heating power is modulated by a phase-control SCR. The SCR is driven by a specialized TCA785 circuit located on the phase control board. The analog control voltage required by TCA785 is provided by the 12-bit digital- to-analog converter MCP4725 that communicates with the digital part of the control system on a two-wire I2C compatible serial interface. The necessary voltage adaptation between the DAC and the phase control circuit is assured through a voltage amplifier based on a rail-to-rail AD820 operational amplifier. The Arduino Uno board is the "brain" of the entire system. It is primarily responsible for the update of the digital control signal u, at every time instance. Therefore, the actual temperature, T_k is read and the actual temperature error (err_k) and change of temperature error ($cerr_k$) are updated, as follows: [8]

$$err_k = T_k - T_{ref}$$

$$cerr_k = err_k - err_{k-1}$$

where err_{k-1} is the temperature in the previous time instant. (1)

The fuzzy logic controller's role is to infer the best modification in the control signal, in every time instance u_k . The digital version of the actual control signal u_k is updated using the relation

$$u_k = u_{k-1} - \Delta u_k \qquad (2)$$

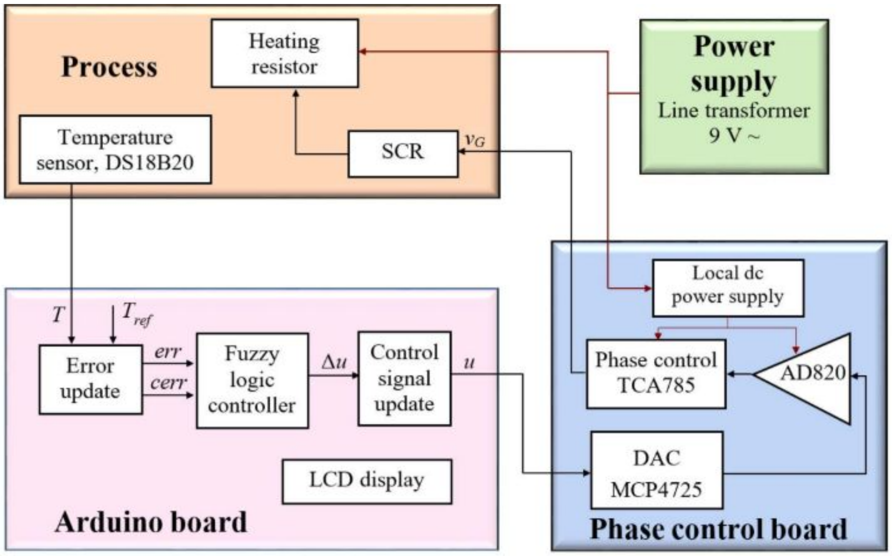


Fig. 4.1: Block diagram of the fuzzy logic-based embedded system for temperature control.

4.2 SYSTEM IMPLEMENTATION

Our proposed system contains two main parts: the power circuit and the control circuit. The power circuit is presented in Fig. 4.2. R is a 2 Ω heating resistor connected to the secondary windings of a line transformer that provides 9 Vrms AC voltage. The mean power dissipated by the resistor is controlled by the Q SCR according with its on/off state. Thus, if the Q is triggered near the start of each positive half-wave (around 0° phase angle) the mean power dissipated by R is near the maximum possible, while if the Q is triggered near the end of each positive half-wave (around 180° phase angle), the dissipated power tends to be zero. The SCR is triggered by the positive pulse of the voltage applied to its gate, v_G, provided by the control circuit.

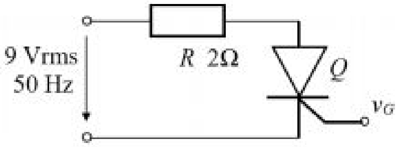


Fig. 4.2: Power Circuit

4.2.1 CONTROL CIRCUIT

The control circuit is presented in Fig. 4.3, and it is developed around the fuzzy logic controller, that has two inputs (*errFls* and *cerrFls*) and one output (*ΔuFls*). The range of value for all these three variables are the same, [-1; 1].

The current temperature error (*err*) and change in temperature error (*cerr)* are determined according to equation (1). For the flexibility of the control system, two scaling factors S_e (for *err*) and S_c (for *cerr*) were introduced. By changing the scaling factor, the user can easily modify the behaviour of the control circuit, the fuzzy logic controller being more or less sensitive to each input; the larger the scaling factor, the bigger the importance of the corresponding variable. To impose upper (+1) and lower (-1) bounds on the input signals, a saturation block is used for each input, so that it clips the signal, when its values surpass the [-1; 1] range.

The fuzzy logic controller generates the *ΔuFls* signal that is further multiplied with the scaling factor S_u obtaining the final modification of the control signal, *Δu* at each time instance. The actual value of the control signal, *u* is than computed according to equation (2). The range of variation for the digital control signal is limited between 0 and 4095 by the saturation block.

The DAC circuit (12 bits) converts the digital control signal to an analog signal, which multiplied by the voltage amplifier, finally generates the control voltage u_a for the phase control specialised circuit.

The phase control circuit (around the TCA785 IC), according with its control voltage u_a, generates the positive voltage pulses v_G that trigger the SCR. It is worth to mention the qualitative relation introduced by the control circuit. For example, if the temperature should increase, *Δu* increases, *u* decreases, u_a decreases, v_G decreases, and finally the mean power dissipated by the heating resistor increases as well, leading to the expected temperature increase.

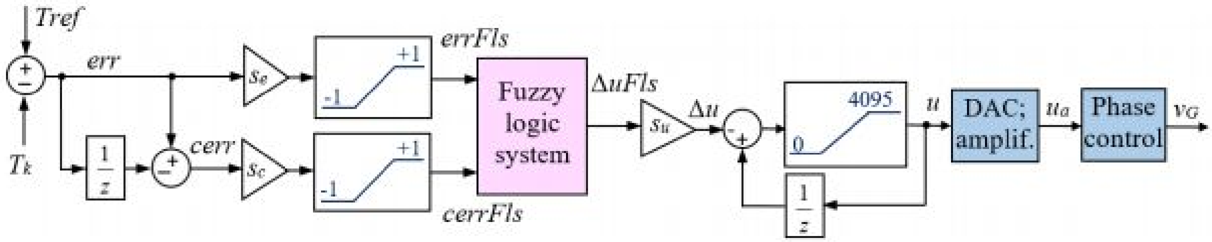


Fig. 4.3: Control Circuit

4.2.2 FUZZY LOGIC CONTROLLER

The system is a first-order Mamdani, with two inputs *errFls* and *cerrFls* and one output *ΔuFls*. For all three variables, the universe of discourse is [-1; 1]. For the inputs, the fuzzy sets are triangular (Fig. 4.4), while for the output fuzzy sets are singleton (Fig. 4.5).

The rule base contains 9 fuzzy rules, represented in Table 1. The defuzzification method, used to transform the partial output fuzzy sets resulted from the inference process into a crisp value is the weighted average method. The control surface that illustrates the full operation of the fuzzy controller is presented in Fig. 4.6

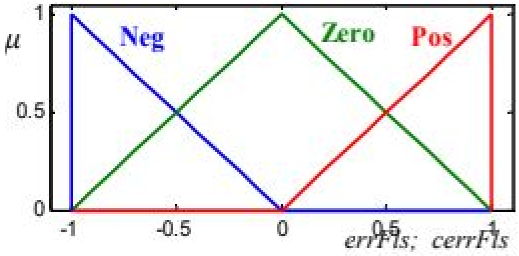


Fig. 4.4: Fuzzy sets for the inputs

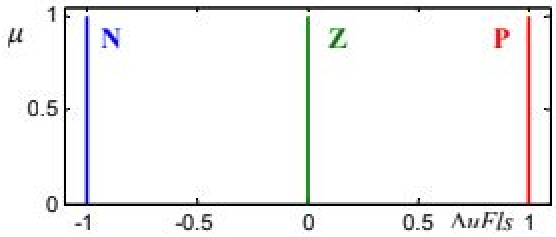


Fig. 4.5: Fuzzy sets for the output

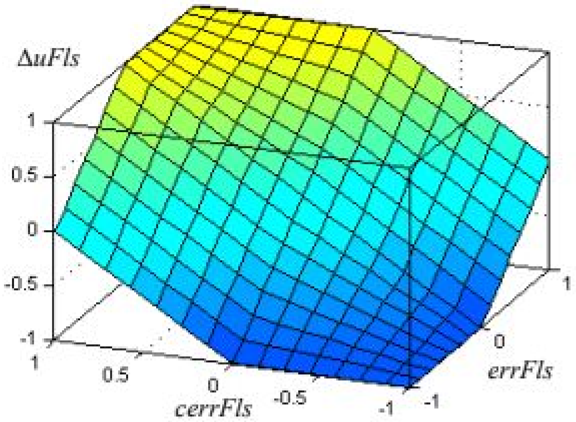


Fig. 4.6: Control surface of the Fuzzy Logic Controller

		errFls		
		Neg	Zero	Pos
cerrFls	Neg	N	N	Z
	Zero	N	Z	P
	Pos	Z	P	P

Table No. 4.1 Rule base of Fuzzy Logic System

Chapter 5
SIMULATION AND RESULTS

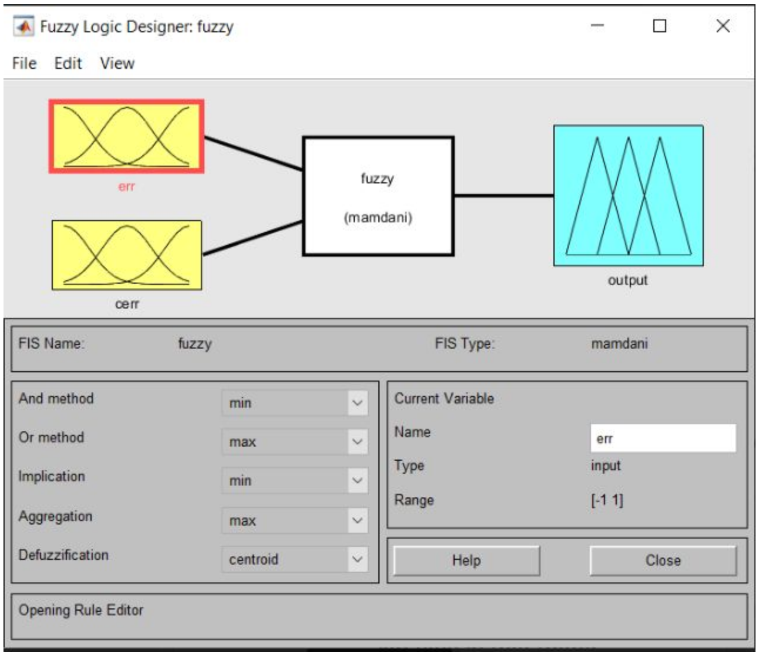
5.1 DS18B20 CONFIGURATION

The DS18B20 digital thermometer provides 9-bit to 12-bit measurements. It senses the temperature and gives the corresponding 9/12 bit binary value, which is given as the input to the Arduino. The Arduino converts the digital input to corresponding temperature value in degrees Celsius. For this purpose, two header add on files namely “One-wire for Arduino” and “Dallas Temperatures” were downloaded and interfaced with Arduino IDE. The code was then run and an object was created in Matlab with the celsius temperature.. This was then converted to a timeseries format, and given as an input to the simulink model.

5.2 FUZZY LOGIC CONTROLLER DESIGN

The fuzzy logic controller as designed using fuzzy toolbox in Matlab. Mamdani inference method was used for the determination of fuzzy rule base. The output of the fuzzy controller is ultimately used to trigger the SCR.

The FLC has two inputs(*errFls* and *cerrrFls*) and one output *ΔuFls*. For all three variables, the universe of discourse is [-1; 1]. For this purpose, saturation blocks were designed.



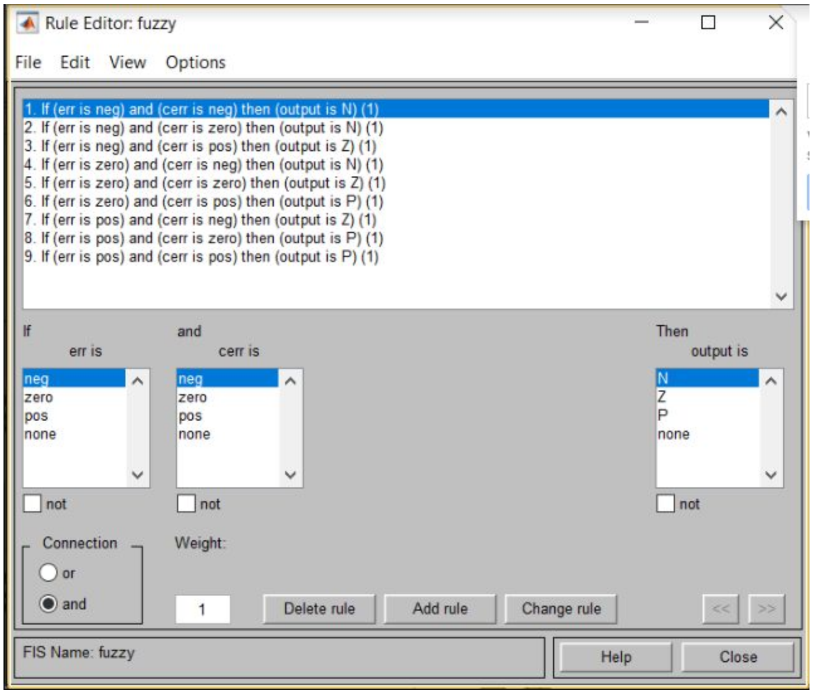
FIS Name: fuzzy
FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: err
Type: input
Range: [-1 1]

Opening Rule Editor

Fig. 5.1: Fuzzy Logic Designer



1. If (err is neg) and (cerr is neg) then (output is N) (1)
2. If (err is neg) and (cerr is zero) then (output is N) (1)
3. If (err is neg) and (cerr is pos) then (output is Z) (1)
4. If (err is zero) and (cerr is neg) then (output is N) (1)
5. If (err is zero) and (cerr is zero) then (output is Z) (1)
6. If (err is zero) and (cerr is pos) then (output is P) (1)
7. If (err is pos) and (cerr is neg) then (output is Z) (1)
8. If (err is pos) and (cerr is zero) then (output is P) (1)
9. If (err is pos) and (cerr is pos) then (output is P) (1)

If err is: neg, zero, pos, none
□ not

and cerr is: neg, zero, pos, none
□ not

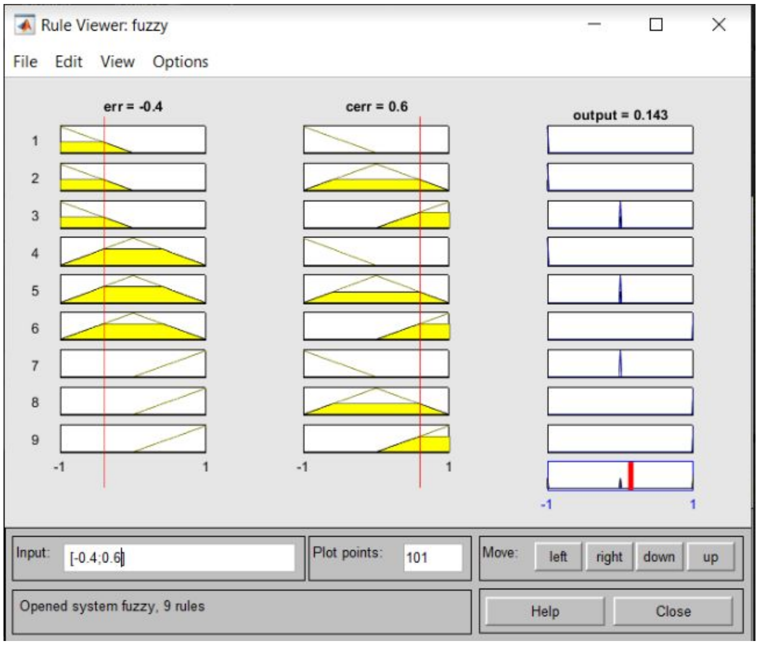
Then output is: N, Z, P, none
□ not

Connection
○ or
● and

Weight: 1

FIS Name: fuzzy

Fig.5.2: Rule Editor



err = -0.4

cerr = 0.6

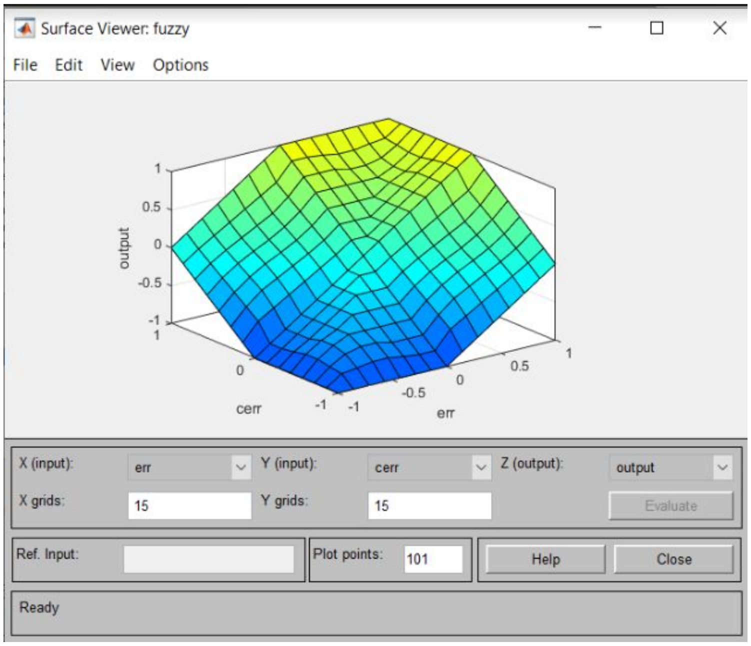
output = 0.143

Input: [-0.4;0.6]

Plot points: 101

Opened system fuzzy, 9 rules

Fig.5.3: Fuzzy Rule Viewer



X (input): err

Y (input): cerr

Z (output): output

X grids: 15

Y grids: 15

Ref. Input:

Plot points: 101

Ready

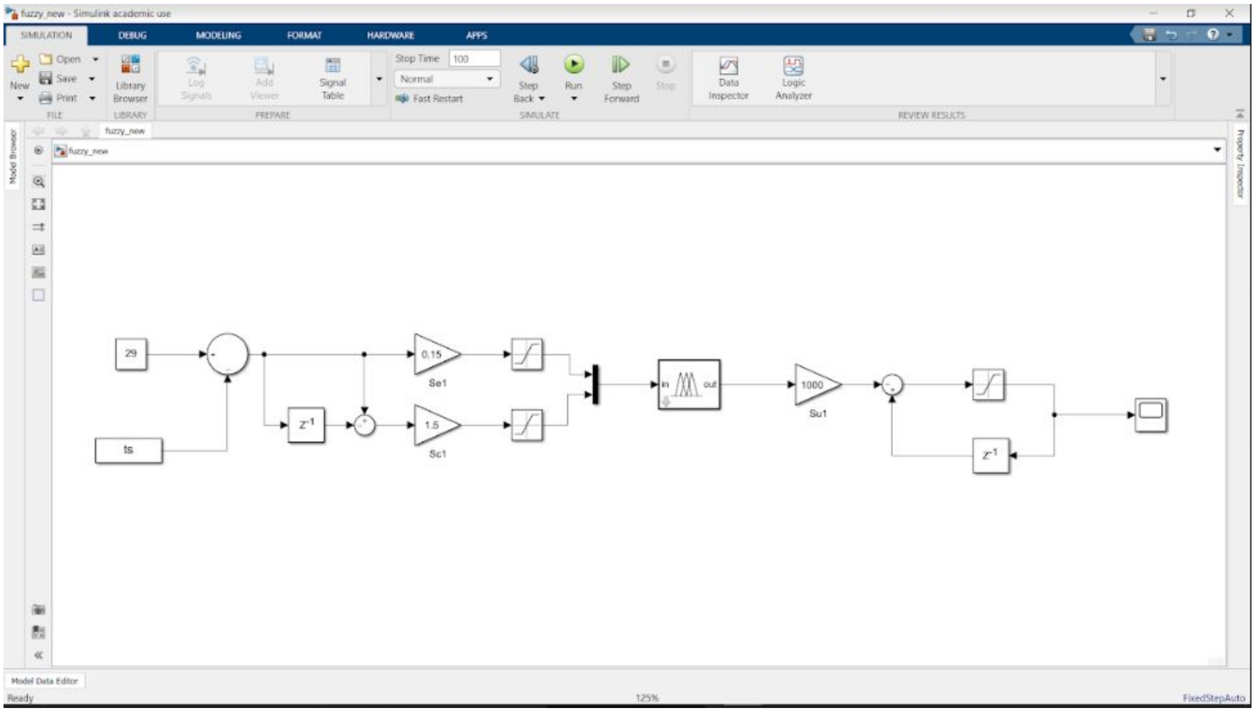
Fig. 5.4: Fuzzy Surface Viewer

5.3 SIMULINK MODEL OF CONTROL CIRCUIT

The control circuit was implemented in simulink. The inputs to the control circuit are *errFls* and *cerrFls,* where

$$err_k = T_k - T_{ref}$$

$$cerr_k = err_k - err_{k-1}$$



Stop Time 100

Normal

Fig. 5.5: Simulink Model

The reference temperature was kept constant at 29°C. It was observed that if the temperature is above the reference temperature, then control signal is a step digital value.. When the measured temperature is below the reference temperature, then the control signal is zero.

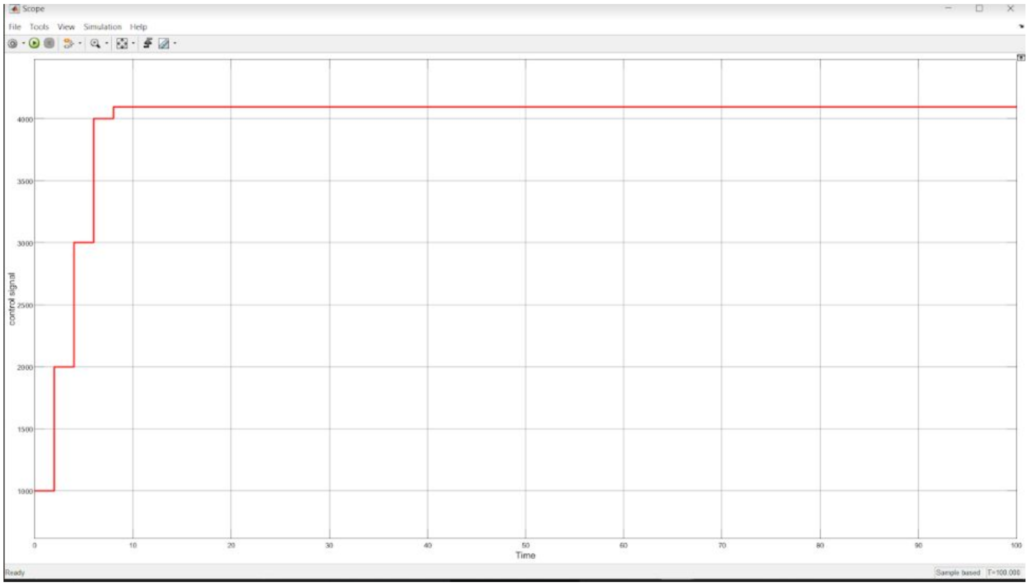


Fig. 5.6: Output when temperature measured is 32^{o}C

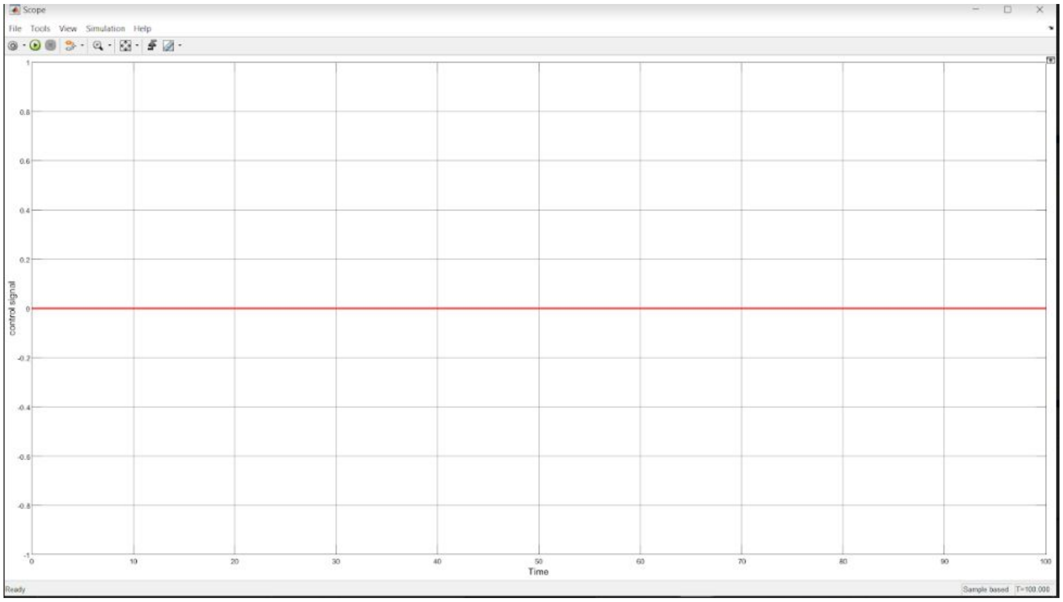


Fig. 5.7: Output when temperature measured is 26^{o}C

Chapter 6
CONCLUSION

The temperature controller aims at maintaining a constant temperature. If the measured temperature is less than reference temperature, then the generated control signal is very less, so that triggering angle of SCR is less, increasing its conduction period, generating more heat due to current flow. Similarly, when the measured temperature is higher than the reference temperature, the control signal is high, reducing the conduction period of SCR.

Care must be taken during temperature measurement due to heat loss by convection and radiation. One wire interface must be achieved properly to avoid data loss between interfaces.

6.1 FUTURE SCOPE

Power circuit is to be implemented for the designed control circuit to maintain temperature at the user defined reference. This will be implemented using heating resistor SCR, phase control IC, precision amplifier and DAC.

Temperature control in ATC70 using the fan embedded in it. The desired temperature is achieved by varying the speed of the fan and heating resistor. This is implemented using a cascaded feedback system using speed as an input parameter.

REFERENCES

[1] https://www.elprocus.com/temperature-controller/

[2]Arduino website, https://www.arduino.cc/

[3] DS18B20 Datasheet, https://goo.gl/ibYqo4

[4]TCA785 Datasheet, https://goo.gl/cF7IdJ

[5]MSP4725 http://ww1.microchip.com/downloads/en/DeviceDoc/22039d.pdf

[6]https://searchenterpriseai.techtarget.com/definition/fuzzy-logic

[7]https://shodhganga.inflibnet.ac.in/bitstream/10603/35998/4/chapter4.pdf

[8]Gabriel Oltean and Laura-Nicoleta Ivanciu, “Implementation of a Fuzzy Logic-Based Embedded System for Temperature Control”, in 2017 40th International Spring Seminar on Electronics Technology (ISSE), IEEE.

APPENDIX

Appendix A:Arduino code for one-wire

```
#include <OneWire.h>
#include <DallasTemperature.h>

// Data wire is connected to the Arduino digital pin 4
#define ONE_WIRE_BUS 10

// Setup a oneWire instance to communicate with any OneWire devices
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature sensor
DallasTemperature sensors(&oneWire);

void setup(void)
{
  sensors.begin();
  Serial.begin(9600);
   sensors.requestTemperatures();

  float temp = sensors.getTempCByIndex(0);
  Serial.println(temp);
  }

void loop(void){
  // Call sensors.requestTemperatures() to issue a global temperature and Requests to all
devices on the bus



}
```

Appendix B:Matlab Code for one-wire

```
a=serial(‘COM3’);
fopen(a);
b=fgets(a);
temp=str2num(b);
ts=timeseries(temp);
```

Appendix C:Fuzzy Controller Code

```
//***********************************************************************
// Matlab .fis to arduino C converter v2.0.1.25122016
// - Karthik Nadig, USA
// Please report bugs to: karthiknadig@gmail.com
//***********************************************************************
#define FIS_TYPE float
#define FIS_RESOLUSION 101
#define FIS_MIN -3.4028235E+38
#define FIS_MAX 3.4028235E+38
typedef FIS_TYPE(*_FIS_MF)(FIS_TYPE, FIS_TYPE*);
typedef FIS_TYPE(*_FIS_ARR_OP)(FIS_TYPE, FIS_TYPE);

typedef FIS_TYPE(*_FIS_ARR)(FIS_TYPE*, int, _FIS_ARR_OP);


//***********************************************************************
// Matlab .fis to arduino C converter v2.0.1.25122016
// - Karthik Nadig, USA
// Please report bugs to:
// https://github.com/karthiknadig/ArduinoFIS/issues
// If you don't have a GitHub account mail to karthiknadig@gmail.com
//***********************************************************************

#include "fis_header.h"

// Number of inputs to the fuzzy inference system
const int fis_gcI = 2;
// Number of outputs to the fuzzy inference system
const int fis_gcO = 1;
// Number of rules to the fuzzy inference system
const int fis_gcR = 9;

FIS_TYPE g_fisInput[fis_gcI];
FIS_TYPE g_fisOutput[fis_gcO];

// Setup routine runs once when you press reset:
void setup()
{
    // initialize the Analog pins for input.
    // Pin mode for Input: error
    pinMode(0 , INPUT);
    // Pin mode for Input: cerror
    pinMode(1 , INPUT);


    // initialize the Analog pins for output.
    // Pin mode for Output: du
    pinMode(2 , OUTPUT);

}

// Loop routine runs over and over again forever:
void loop()
{
    // Read Input: error
    g_fisInput[0] = analogRead(0);
    // Read Input: cerror
    g_fisInput[1] = analogRead(1);

    g_fisOutput[0] = 0;

    fis_evaluate();

    // Set output vlaue: du
    analogWrite(2 , g_fisOutput[0]);

}

//***********************************************************************
// Support functions for Fuzzy Inference System
//***********************************************************************
// Triangular Member Function
FIS_TYPE fis_trimf(FIS_TYPE x, FIS_TYPE* p)
{
    FIS_TYPE a = p[0], b = p[1], c = p[2];
    FIS_TYPE t1 = (x - a) / (b - a);
    FIS_TYPE t2 = (c - x) / (c - b);
```

```
   if ((a == b) && (b == c)) return (FIS_TYPE) (x == a);
   if (a == b) return (FIS_TYPE) (t2*(b <= x)*(x <= c));
   if (b == c) return (FIS_TYPE) (t1*(a <= x)*(x <= b));
   t1 = min(t1, t2);
   return (FIS_TYPE) max(t1, 0);
}

FIS_TYPE fis_min(FIS_TYPE a, FIS_TYPE b)
{
   return min(a, b);
}

FIS_TYPE fis_max(FIS_TYPE a, FIS_TYPE b)
{
   return max(a, b);
}

FIS_TYPE fis_array_operation(FIS_TYPE *array, int size, _FIS_ARR_OP pfnOp)
{
   int i;
   FIS_TYPE ret = 0;

   if (size == 0) return ret;
   if (size == 1) return array[0];

   ret = array[0];
   for (i = 1; i < size; i++)
   {
      ret = (*pfnOp)(ret, array[i]);
   }

   return ret;
}


//***********************************************************************
// Data for Fuzzy Inference System
//***********************************************************************
// Pointers to the implementations of member functions
```

```
_FIS_MF fis_gMF[] =
{
   fis_trimf
};

// Count of member function for each Input
int fis_gIMFCount[] = { 3, 3 };

// Count of member function for each Output
int fis_gOMFCount[] = { 3 };

// Coefficients for the Input Member Functions
FIS_TYPE fis_gMFI0Coeff1[] = { -2, -1, 0 };
FIS_TYPE fis_gMFI0Coeff2[] = { -1, 0, 1 };
FIS_TYPE fis_gMFI0Coeff3[] = { 0, 1, 2 };
FIS_TYPE* fis_gMFI0Coeff[] = { fis_gMFI0Coeff1, fis_gMFI0Coeff2, fis_gMFI0Coeff3 };
FIS_TYPE fis_gMFI1Coeff1[] = { -2, -1, 0 };
FIS_TYPE fis_gMFI1Coeff2[] = { -1, 0, 1 };
FIS_TYPE fis_gMFI1Coeff3[] = { 0, 1, 2 };
FIS_TYPE* fis_gMFI1Coeff[] = { fis_gMFI1Coeff1, fis_gMFI1Coeff2, fis_gMFI1Coeff3 };
FIS_TYPE** fis_gMFICoeff[] = { fis_gMFI0Coeff, fis_gMFI1Coeff };

// Coefficients for the Output Member Functions
FIS_TYPE fis_gMFO0Coeff1[] = { -1, -1, -1 };
FIS_TYPE fis_gMFO0Coeff2[] = { 0, 0, 0 };
FIS_TYPE fis_gMFO0Coeff3[] = { 1, 1, 1 };
FIS_TYPE* fis_gMFO0Coeff[] = { fis_gMFO0Coeff1, fis_gMFO0Coeff2, fis_gMFO0Coeff3 };
FIS_TYPE** fis_gMFOCoeff[] = { fis_gMFO0Coeff };

// Input membership function set
int fis_gMFI0[] = { 0, 0, 0 };
int fis_gMFI1[] = { 0, 0, 0 };
int* fis_gMFI[] = { fis_gMFI0, fis_gMFI1};

// Output membership function set
int fis_gMFO0[] = { 0, 0, 0 };
int* fis_gMFO[] = { fis_gMFO0};

// Rule Weights
```

```
FIS_TYPE fis_gRWeight[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1 };

// Rule Type
int fis_gRType[] = { 1, 1, 1, 1, 1, 1, 1, 1, 1 };

// Rule Inputs
int fis_gRI0[] = { 1, 1 };
int fis_gRI1[] = { 2, 1 };
int fis_gRI2[] = { 3, 1 };
int fis_gRI3[] = { 1, 2 };
int fis_gRI4[] = { 2, 2 };
int fis_gRI5[] = { 3, 2 };
int fis_gRI6[] = { 1, 3 };
int fis_gRI7[] = { 2, 3 };
int fis_gRI8[] = { 3, 3 };
int* fis_gRI[] = { fis_gRI0, fis_gRI1, fis_gRI2, fis_gRI3, fis_gRI4, fis_gRI5, fis_gRI6,
fis_gRI7, fis_gRI8 };

// Rule Outputs
int fis_gRO0[] = { 1 };
int fis_gRO1[] = { 1 };
int fis_gRO2[] = { 2 };
int fis_gRO3[] = { 1 };
int fis_gRO4[] = { 2 };
int fis_gRO5[] = { 3 };
int fis_gRO6[] = { 2 };
int fis_gRO7[] = { 3 };
int fis_gRO8[] = { 3 };
int* fis_gRO[] = { fis_gRO0, fis_gRO1, fis_gRO2, fis_gRO3, fis_gRO4, fis_gRO5, fis_gRO6,
fis_gRO7, fis_gRO8 };

// Input range Min
FIS_TYPE fis_gIMin[] = { -1, -1 };

// Input range Max
FIS_TYPE fis_gIMax[] = { 1, 1 };

// Output range Min
FIS_TYPE fis_gOMin[] = { -1 };
```

```
// Output range Max
FIS_TYPE fis_gOMax[] = { 1 };

//***********************************************************************
// Data dependent support functions for Fuzzy Inference System
//***********************************************************************
FIS_TYPE fis_MF_out(FIS_TYPE** fuzzyRuleSet, FIS_TYPE x, int o)
{
    FIS_TYPE mfOut;
    int r;

    for (r = 0; r < fis_gcR; ++r)
    {
        int index = fis_gRO[r][o];
        if (index > 0)
        {
            index = index - 1;
            mfOut = (fis_gMF[fis_gMFO[o][index]])(x, fis_gMFOCoeff[o][index]);
        }
        else if (index < 0)
        {
            index = -index - 1;
            mfOut = 1 - (fis_gMF[fis_gMFO[o][index]])(x, fis_gMFOCoeff[o][index]);
        }
        else
        {
            mfOut = 0;
        }

        fuzzyRuleSet[0][r] = fis_min(mfOut, fuzzyRuleSet[1][r]);
    }
    return fis_array_operation(fuzzyRuleSet[0], fis_gcR, fis_max);
}

FIS_TYPE fis_defuzz_centroid(FIS_TYPE** fuzzyRuleSet, int o)
{
    FIS_TYPE step = (fis_gOMax[o] - fis_gOMin[o]) / (FIS_RESOLUSION - 1);
    FIS_TYPE area = 0;
    FIS_TYPE momentum = 0;
    FIS_TYPE dist, slice;
    int i;

    // calculate the area under the curve formed by the MF outputs
    for (i = 0; i < FIS_RESOLUSION; ++i){
        dist = fis_gOMin[o] + (step * i);
        slice = step * fis_MF_out(fuzzyRuleSet, dist, o);
        area += slice;
        momentum += slice*dist;
    }

    return ((area == 0) ? ((fis_gOMax[o] + fis_gOMin[o]) / 2) : (momentum / area));
}

//***********************************************************************
// Fuzzy Inference System
//***********************************************************************
void fis_evaluate()
{
    FIS_TYPE fuzzyInput0[] = { 0, 0, 0 };
    FIS_TYPE fuzzyInput1[] = { 0, 0, 0 };
    FIS_TYPE* fuzzyInput[fis_gcI] = { fuzzyInput0, fuzzyInput1, };
    FIS_TYPE fuzzyOutput0[] = { 0, 0, 0 };
    FIS_TYPE* fuzzyOutput[fis_gcO] = { fuzzyOutput0, };
    FIS_TYPE fuzzyRules[fis_gcR] = { 0 };
    FIS_TYPE fuzzyFires[fis_gcR] = { 0 };
    FIS_TYPE* fuzzyRuleSet[] = { fuzzyRules, fuzzyFires };
    FIS_TYPE sW = 0;

    // Transforming input to fuzzy Input
    int i, j, r, o;
    for (i = 0; i < fis_gcI; ++i)
    {
        for (j = 0; j < fis_gIMFCount[i]; ++j)
        {
            fuzzyInput[i][j] =
                (fis_gMF[fis_gMFI[i][j]])(g_fisInput[i], fis_gMFICoeff[i][j]);
        }
}

int index = 0;
for (r = 0; r < fis_gcR; ++r)
{
   if (fis_gRType[r] == 1)
   {
      fuzzyFires[r] = FIS_MAX;
      for (i = 0; i < fis_gcI; ++i)
      {
         index = fis_gRI[r][i];
         if (index > 0)
            fuzzyFires[r] = fis_min(fuzzyFires[r], fuzzyInput[i][index - 1]);
         else if (index < 0)
            fuzzyFires[r] = fis_min(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
         else
            fuzzyFires[r] = fis_min(fuzzyFires[r], 1);
      }
   }
   else
   {
      fuzzyFires[r] = FIS_MIN;
      for (i = 0; i < fis_gcI; ++i)
      {
         index = fis_gRI[r][i];
         if (index > 0)
            fuzzyFires[r] = fis_max(fuzzyFires[r], fuzzyInput[i][index - 1]);
         else if (index < 0)
            fuzzyFires[r] = fis_max(fuzzyFires[r], 1 - fuzzyInput[i][-index - 1]);
         else
            fuzzyFires[r] = fis_max(fuzzyFires[r], 0);
      }
   }

   fuzzyFires[r] = fis_gRWeight[r] * fuzzyFires[r];
   sW += fuzzyFires[r];
}

if (sW == 0)
```

```
    {
        for (o = 0; o < fis_gcO; ++o)
        {
            g_fisOutput[o] = ((fis_gOMax[o] + fis_gOMin[o]) / 2);
        }
    }
    else
    {
        for (o = 0; o < fis_gcO; ++o)
        {
            g_fisOutput[o] = fis_defuzz_centroid(fuzzyRuleSet, o);
        }
    }

}
```