Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas

upiita-ipn

Neurofuzzy Systems

Fuzzy Logic Project
Silent traffic lights

Team members:

Cano Zapata Santiago

Costilla Caballero Salvador

Domínguez Durán Pablo Melquiades

Elizarraras Llanos Angel Gustavo

Ramírez Marín Julián Andrés

Professor: Tovar Corona Blanca

Due date: 27/03/20

Semester 20/2

INDEX

Introduction

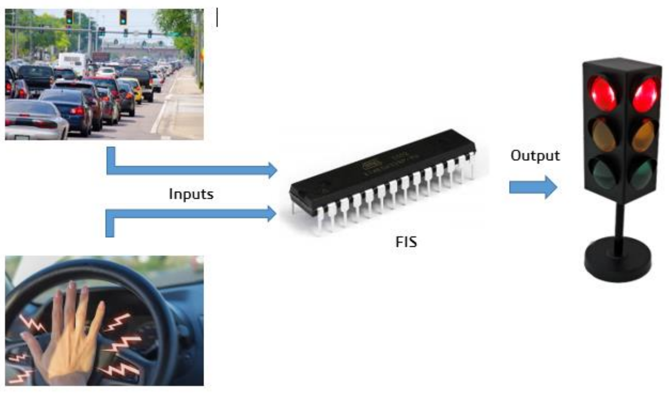
In this project we present a solution idea to noise pollution due to claxons honks in the waiting of traffic lights turns. It consists of a patience reward/punishment system for drivers.
The red-light waiting time is proposed to be the output of our system. The time it stays on depends on how much noise our sensor detects and how many cars there are waiting. In the contents of this report, we talk furthermore about the procedure and the hardware used and code implementation for the development of this idea.

Problem to solve

Nowadays, with overpopulation in big cities, pollution has become a big part of our daily lives and it comes in different ways, such as water, air, soil, light pollution, but as a topic of matter, we are interested in noise pollution which is unhealthy for human beings because unwanted sound (noise) is proved to damage mental health. It can also cause hypertension, high stress levels, tinnitus, hearing loss, sleep disturbances, and other harmful and disturbing effects.
When being in peak hours, is very common to hear the claxon honks of cars nearby, it is annoying, irritating and useless because the traffic will not go away and the drivers can't go faster. So, our project idea is a traffic light system that rewards patient people, mainly when there is too much noise caused by cars, so the red light will stay more time; also, it will depend on how many cars are waiting for the green light.

Mamdani FIS

To be a little bit more illustrative, this is our Mamdani FIS.



Objective

Apply our knowledge in fuzzy logic to build a prototype of an inference system that prevent noise pollution produced by claxons based in a reward/punishment system.

Materials

Arduino UNO

Arduino is an open-source prototyping platform used for building electronics projects. It consists of both a physical programmable circuit board and its own IDE, so it is simple and easy to implement. Also, it has analog and digital inputs, which is very useful for us because we need both, that is why we choose it for controlling our system.

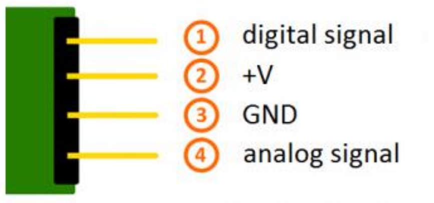
Buttons

They are normally-open tactile switches and we will use them to simulate if there are cars or not.

KY-037 Sound Detection Sensor

The sensor has 3 main components on its circuit board. First, the sensor unit at the front of the module which measures the area physically and sends an analog signal to the second unit, the amplifier. The amplifier amplifies the signal, according to the resistant value of the potentiometer, and sends the signal to the analog output of the module.

The third component is a comparator which switches the digital out and the LED if the signal falls under a specific value. You can control the sensitivity by adjusting the potentiometer.



Buzzer

A buzzer or beeper is an audio signaling device, which may be mechanical, electromechanical, or piezoelectric, we are going to simulate the noise with it.

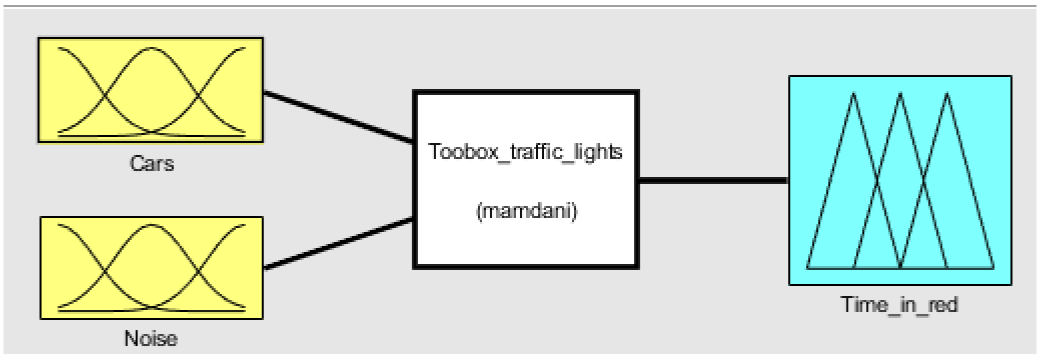
Leds

We will use them to simulate the traffic light, which is the output.

Development

Design of the FIS

The next image is the overall system given by the toolbox. Like we said, the inputs are the noise by the claxons and the number of cars waiting for the green light, and depending on these, the red light waiting time is going to increase or decrease. This system is designed for only one way of the road.

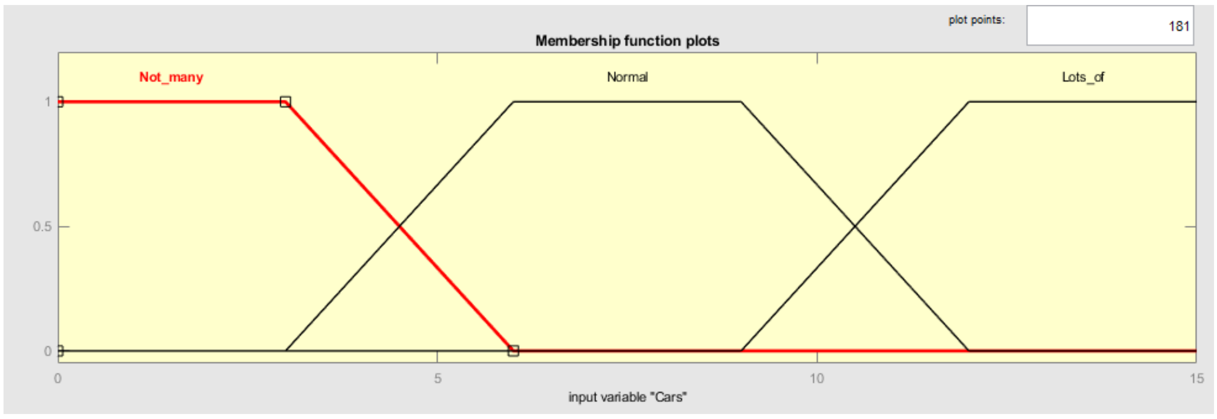


Definition of the inputs and outputs

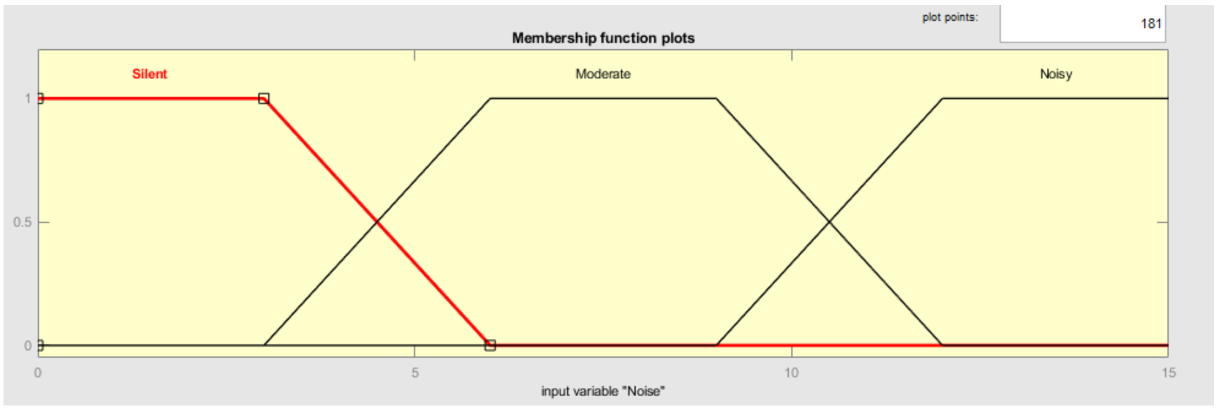
As we need the same length in the input universes, we make the noise dependable on the actual number of cars. Our noise sensor give us a voltage range, that we escalate depending on our membership function. For the vehicles, we have push buttons that represents certain numbers (weight values) for each member of the function.

Membership Functions

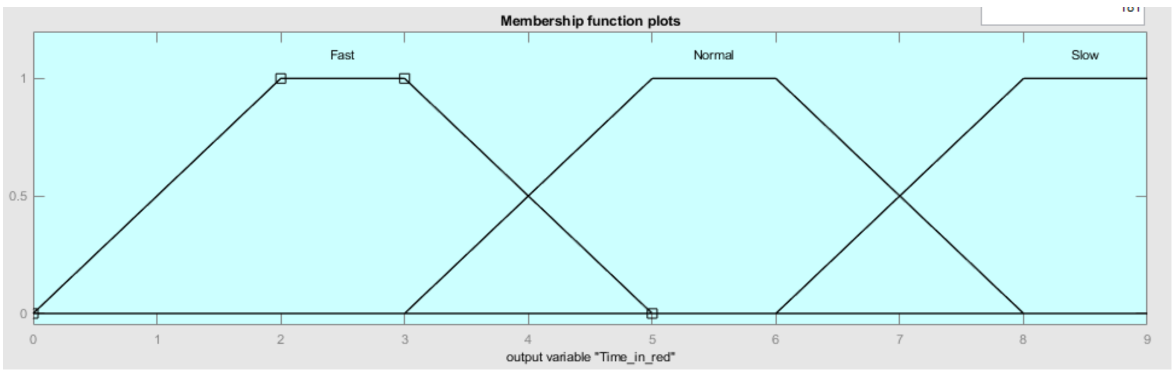
For all our membership functions we used trapezoidal ones. For the cars, the first MF, “Not_many” starts on 1 because when there are few cars, it has to be a priority for the system, the same goes when we have too many cars. And for the MF left, we tried to make it symmetrical.



Because we make the noise scalable depending on the cars, it has the same logic and MF than the cars.



Finally, for the output, the red light time, we decided for that range because it is a fast demonstration of the system and too much time would be tedious. Also, it would not be possible for a traffic light to always be in green, so the first MF “Fast”, starts in 0, so it would not give a 0 value and for “Slow” it keeps in the end in an 1 value, so a high value is possible.



Inference table

Noise / Cars	Not many	Normal	Lots of
Noisy	Normal	Slow	Slow
Moderate	Normal	Normal	Normal
Silent	Fast	Fast	Normal

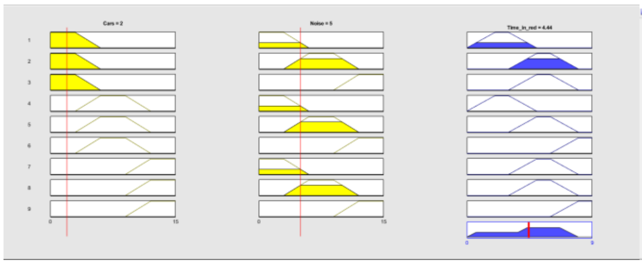
Inference Rules

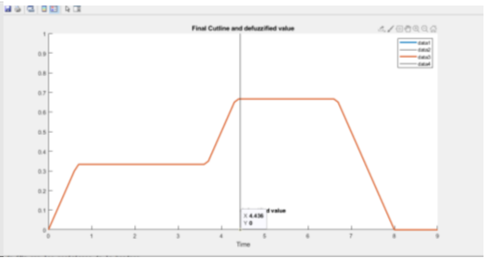
If cars	not many	and noise	silent	then time	fast
If cars	not many	and noise	moderate	then time	normal
If cars	not many	and noise	noisy	then time	normal
If cars	normal	and noise	silent	then time	fast
If cars	normal	and noise	moderate	then time	normal
If cars	normal	and noise	noisy	then time	slow
If cars	lots of	and noise	silent	then time	normal
If cars	lots of	and noise	moderate	then time	normal
If cars	lots of	and noise	noisy	then time	slow

Results

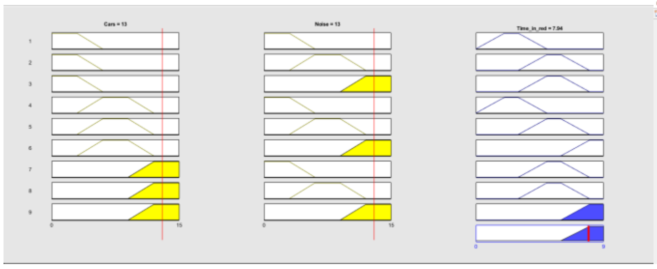
Examples and comparison with the Toolbox and the Algorithm

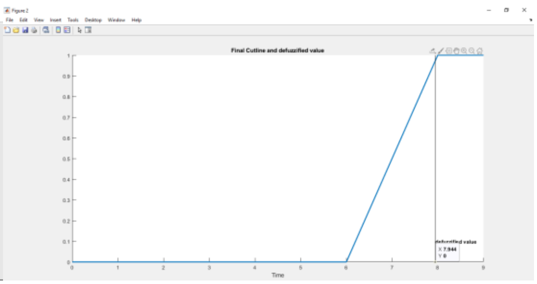
In this example, there are 2 cars and a level 5 of noise, it give us a time in red of 4.44 in both, the algorithm and the toolbox.



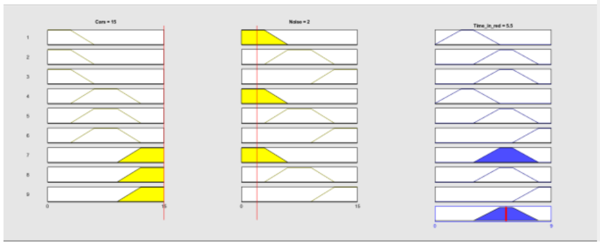


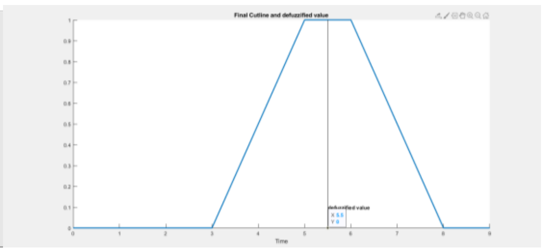
In this example we can notice the case of “punishment” when there are a lot of cars (13) and too much noise (13), the time they are going to wait is high.





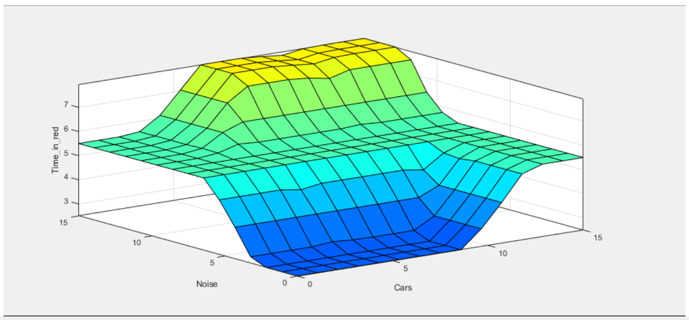
And this case corresponds to the “reward”, there are too many cars (15) but they are in silent (2), so the time is not that long (5.5 seconds).

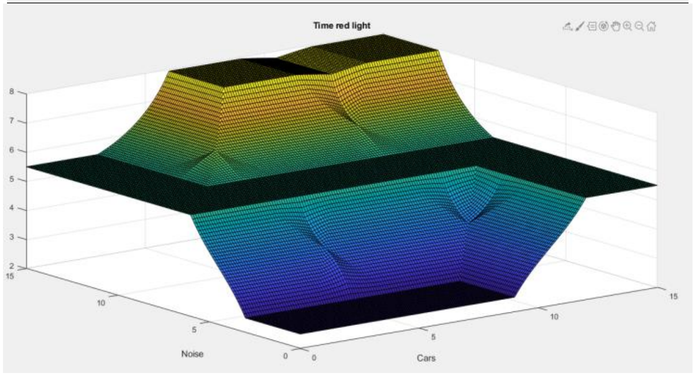




3D graphs

Both graphs are the same, but in the algorithm, we put too many steps in the universes to watch the curves more accurate**.**





Connection to Arduino

In MATLAB you can create an arduino object that represents a connection to Arduino and then we can program our hardware with MATLAB code.
The sintax and description is this:

`a = arduino(port,board)` creates a connection to Arduino hardware on the specified serial port.

Then we call the FIS created and give it the input data we receive with the arduino, and then, the output data goes from the FIS to the hardware.

Working Prototype

https://youtu.be/cFk46lNl5Co

Conclusions

Overall, the implementation of our project idea and system was successful. At first, we had problems defining the membership functions and how we were going to implement our FIS in the Arduino board.

In collaboration we proposed ideas for the better implementation of the MF and we selected which one was the more accurate for our proposal. Then, we looked for how to connect Arduino to MATLAB, that later result in some more problems trying to read the data of the inputs, other problems when reading the Serial Port and writing data to the Arduino, but at the end we sorted it out for good.

This project helped us improving our teamwork skills and our knowledge in the electronics field, which led us to a far more understanding of the Mamdani Fuzzy Inference System method and Fuzzy Logic.

References

- https://www.youtube.com/watch?v=LWXMI_xSyPI
- Professor Blanca Tovar Corona notes and practices.
- https://www.mathworks.com/help/supportpkg/arduinoio/ref/arduino.html

Anexes

As it can be consulted in the repository:

https://github.com/pablomdd/Traffic-Light-Noise-Punishment-System/tree/master

Code for Arduino

```
clear all;

%Inicializacion del Arduino

% PINES
%-----ENTRADAS---
%  SONIDO


%   NUMERO DE COCHES (DIP SWITCH)
%   dipPins[] = {2, 3, 4, 5}; DIGITALES

%-----SALIDAS----
%D11    LED ROJO
%D12    LED VERDE

sisFuzzy = readfis('Toobox_traffic_lights.fis');

%Calibracion

%Inicio del Programa
delete(instrfind({'port'},{'COM7'}));
a=arduino('COM7','uno');
disp('Este programa fue desarrollado con el fin de:');
disp('Si pitas, semaforo rojo');
disp('Proyecto Logica difusa');
pause(5)
while(true)
    %Semaforo inicia en verde

    disp('Lectura de datos');
    pause(2);
    writeDigitalPin(a, 'D2', 0);
    writeDigitalPin(a, 'D3', 1);
    pause(4)
    writeDigitalPin(a, 'D3', 0);
    writeDigitalPin(a,'D4',1);
    pause(2)
    writeDigitalPin(a,'D4',0);
    %PULSADORES SONIDO
    estadoB=readDigitalPin(a,'D8');
    estadoC=readDigitalPin(a,'D9');
    estadoD=readDigitalPin(a,'D10');
    %ESTADO B

    estadoB=17
    Sensor_sonido=readVoltage(a,'A0');
    Sensor_sonido=Sensor_sonido*5
    [tiempoSemaforo] =
evalfis([estadoB,Sensor_sonido],sisFuzzy)
    writeDigitalPin(a, 'D2', 1);
    writeDigitalPin(a, 'D3', 0);
    pause(tiempoSemaforo);

    writeDigitalPin(a, 'D2', 0);
    writeDigitalPin(a, 'D3', 1);
    pause(4)
    writeDigitalPin(a, 'D3', 0);
    writeDigitalPin(a,'D4',1);
    pause(2)
    writeDigitalPin(a,'D4',0);

    %ESTADO C
    estadoC=8
    Sensor_sonido=readVoltage(a,'A0');
    Sensor_sonido=Sensor_sonido*1.5
    [tiempoSemaforo] =
evalfis([estadoC,Sensor_sonido],sisFuzzy)
    writeDigitalPin(a, 'D2', 1);
    writeDigitalPin(a, 'D3', 0);
    pause(tiempoSemaforo);
    pause(1)

    writeDigitalPin(a, 'D2', 0);
    writeDigitalPin(a, 'D3', 1);
    pause(4)
    writeDigitalPin(a, 'D3', 0);
    writeDigitalPin(a,'D4',1);
    pause(2)
    writeDigitalPin(a,'D4',0);

    %ESTADO D
    estadoD=2
    Sensor_sonido=readVoltage(a,'A0')
    Sensor_sonido=Sensor_sonido-0.7;
    [tiempoSemaforo] =
evalfis([estadoD,Sensor_sonido],sisFuzzy)
    writeDigitalPin(a, 'D2', 1);
    writeDigitalPin(a, 'D3', 0);
    pause(tiempoSemaforo);
    pause(1)

    %Se manda a llamar el Toolbox
    % Semaforo se pone en Rojo

end
```

“Main” code

```
u1 = 0 :0.1: 15;

TR = 0 :0.1:9;

LOW = Trapezo(TR,6,8,9,9);
MED = Trapezo(TR, 3,5,6,8);
HIGH = Trapezo(TR, 0, 2, 3, 5);



[X, Y] = meshgrid(0:0.1:15, 0:0.1:15);
Zs = X;
u2 = 0:0.1:15;
x = 1;
for i = u2
    y = 1;
    for j = u2
        [FCLTR] = Codeo2(u2, i, j);
        Tiempor = defuzz(TR, FCLTR, 'centroid');
        Zs(y, x) = Tiempor;
        y = y + 1;
    end
    x = x + 1;
end

figure(1)
surf(X, Y, Zs)
title('Time red light')
xlabel('Cars')
ylabel('Noise')
```

```
    val1 = input('Introduce value for cars: ');
    val2 = input('Introduce value for noise: ');



    [FCLTR] = Codeo2(u1, val1, val2);

    figure(2)
    hold on
    plot(TR, FCLTR, 'LineWidth', 2);
    title('Final Cutline and defuzzified value')
    xlabel('Time')


    Time_red = defuzz(TR, FCLTR, 'centroid');

    h2 = line([Time_red Time_red],[0 1],'Color','k');
    t2 = text(Time_red,0.1,'defuzzified
value','FontWeight','bold');
    hold on

    fprintf('Time of the red light: %.2f \n', Time_red);
```

Algorithm function

```
function [FCLTR] = Codeo2(u, val1, val2)

%% CARS
    PC = Trapezo(u, 0, 0, 3, 6);
    IC = Trapezo(u, 3, 6, 9, 12);
    MC = Trapezo(u, 9, 12, 15, 15);

%% RUIDO
    PO = Trapezo(u, 0, 0, 3, 6);
    MO = Trapezo(u, 3, 6, 9, 12);
    RO = Trapezo(u, 9, 12, 15, 15);



%% TIME RED LIGHT
    TR = 0 :0.1:9;
    L = Trapezo(TR,6,8,9,9);
    M = Trapezo(TR, 3,5,6,8);
    H = Trapezo(TR, 0, 2, 3, 5);

    SEPC = PC(u == val1);
    SEIC = IC(u == val1);
    SEMC = MC(u == val1);

    vectorcoches=[SEPC,SEIC,SEMC];



    SEPO = PO(u == val2);
    SEMO = MO(u == val2);
    SERO = RO(u == val2);

    vectorruido=[SERO,SEMO,SEPO];
    op=[];
    k=1;
    for i=1:3
    for j=1:3
        op(k)=min(vectorcoches(i),vectorruido(j));
        k=k+1;
    end
end

    clL=min(L,max([op(4),op(7)]));
    clM=min(M,max([op(1),op(2),op(5),op(8),op(9)]));
    clH=min(H,max([op(3),op(6)]));
    FCLTR=max(clL,max(clM,clH));


end
```

Trapezoidal Membership Function

```
function [y] = mf_trapezoidal(x, a, b, c, d)
%     start = x(1);
%     step = x(2);
%     stop = x(3);
%     x = start : step : stop;

    y(x<a)= 0;

    x2 = x(a <= x & x < b);
    y(a <= x & x < b)= (x2 - a) / (b - a);

    y(b <= x & x < c) = 1;

    x4 = x(c <= x & x <= d);
    y(c <= x & x <= d) = (d - x4) / (d - c);

    y(d < x) = 0;

    if isnan(y(length(y)))
        y(length(y)) = 1;
    end
end



end
```