UNIVERSITY OF COLIMA
Smart Car Race

Students: **Francisco Javier Batista Vargas**,
Juan Manuel Gonzalez Rodriguez,
Gabriel Navarro Marquez.
Teacher: **Ramon Antonio Felix Cuadras,**
Jose Rodolfo Madrigal Sanchez.

November 2, 2009

Abstract

In this report we discuss the SMART CAR, which was designed to participate in the competition line follower meeting the rules and parameters set for the competition. Besides the hardware necessary to sense, he adapted a speed control based on PWM technique, implemented in a microcontroller HC12 family.

The Smart car is a car line follower consists of the electronics with a microprocessor HC12 family, which is responsible for controlling the car through the interpretation of the signals received from sensors CNY70, which will track a black line marked on a white track or platform, that road must be traveled in the shortest possible time, in addition to turn and climb a particular slope.

1 Description of Major Concepts and Specific Technical Implementation Schemes for Design and Production of Car Models

The major angles that form a vehicle are as follows;

Drop angle: The angle between the vertical wheel with the ground. Its function is to improve grip in the corners to compensate for the deformation of the tire lateral force to do it.

Caster angle: The angle that we see when we look at the vehicle side, the angle of the knuckle - or the tube with the vertical direction (this is usually measured on the bikes) or with the ground (on bikes).

Departure angle: Looking at the vehicle in front, is the angle between the pivot to the vertical direction.

Beam angle: It is known as convergence and divergence. If we look at a car from above, is the angle between the wheels with the central straight line of the vehicle. At first glance it seems logical that the 4 wheels perfectly parallel is the best solution. In line, 4-wheel parallel produce as little friction as possible.

Ackerman angle: The Ackerman geometry suggests that when a vehicle rounds a curve, the inner wheel rotates around a smaller circle than the outside wheel. This is evident because of the width of the vehicle. Therefore, for any wheel slippage while turning, the inner wheel must turn slightly steeper angle than the outside. This geometry is used in production cars and is more comfortable to drive and minimizing tire wear.

2 Description of Mechanical Design of Car Model

The mechanical design of the smart car has developed under the rules of competition, this being a fan of online shopping by sensors, so this need because it has had to generate the necessary forms for mechanical design.

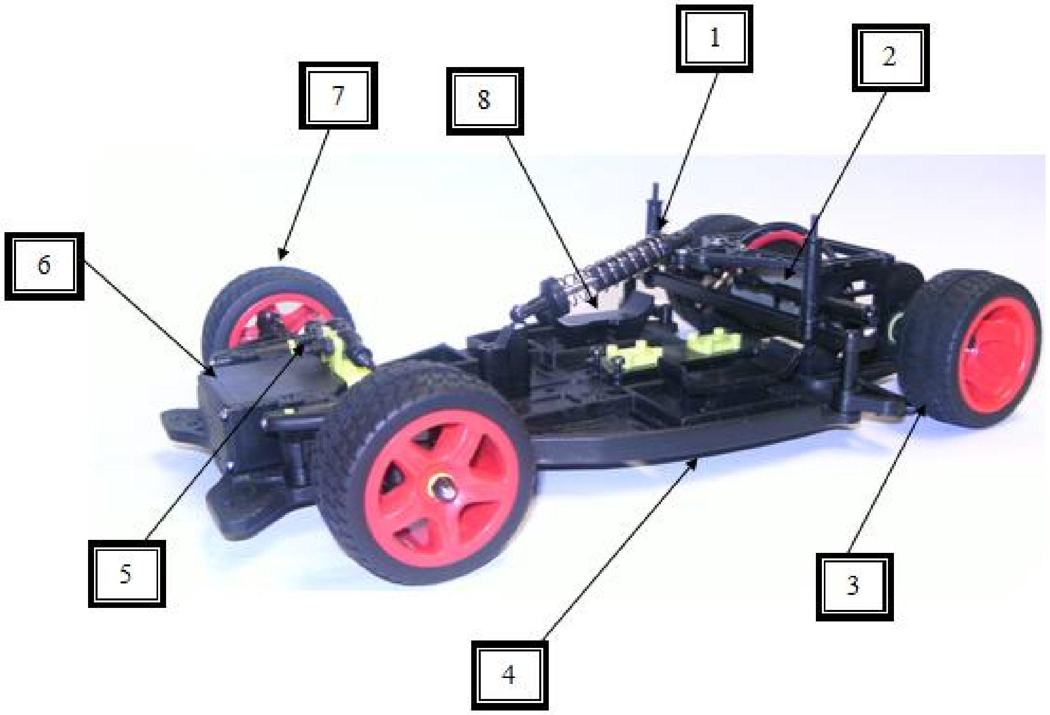

Fig. 1. Smart Car Model

Model description

1. Suspension
2. Engine Friction
3. Rear Axle
4. Chassis
5. Mechanism Ackerman
6. Servomotor
7. Front Axle
8. Battery holder

Once you have defined the model of the plant, either by using ID Systems or any other modeling technique, we proceed to create the control structure and generating the same parameters for a given design specifications.

3 Description of Control Circuit Design

"As the complexity of a system increase, our ability to be accurate and build instructions about their behavior diminishes until a threshold beyond which the accuracy and meaning are mutually exclusive characteristics."

One of the disciplines of mathematics with the greatest number of followers now is called fuzzy logic or fuzzy, which is the logic that uses expressions that are neither completely true nor completely false, that is, the logic applied to concepts that can take a value any truth in a set of values ranging between two extremes, the absolute truth and complete falsehood. Should be emphasized that what is diffuse, fuzzy, imprecise or vague fuzzy logic is not each other, but the object that studies and expresses the lack of definition of the concept to which it applies. Fuzzy logic allows imprecise process information such as average height or low temperature, in terms of fuzzy sets that are combined into rules to define actions: if the temperature is high then fervid. Thus control systems based on fuzzy combination of input variables, defined in terms of fuzzy sets, using sets of rules that produce one or more output values.

3.1 System Block Diagram Technique Based on Fuzzy Logic

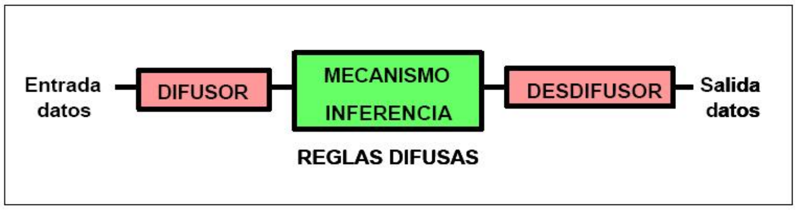

Fig. 2. General Scheme of a system based on fuzzy logic.

Description of each of the blocks.

Block diffuser; as we can see in the figure is the block in which each input variable is assigned a grade of membership to each fuzzy sets has been considered by the characteristic functions associated to these fuzzy sets. The inputs to this block are specific values of input variables and outputs are membership degrees of fuzzy sets considered.

Block Inference; as shown in the figure is the intermediate block between input and output, this block contains the inference mechanisms to be discussed later, relates fuzzy sets of input and output and representing the rules that define the system. The inputs to this block are fuzzy sets (degrees of membership) and the output fuzzy sets are also associated with the output variable.

Block Desdifusor: is the final block of the overall scheme based on fuzzy logic block from the fuzzy set obtained in the inference mechanism and through desdifusion mathematical methods, yields a specific value of the variable data output, is ie the result.

3.2 Control System Used in the Smart Car

A dynamic system can be defined conceptually as an entity that receives some external actions or input variables, and whose answer to these external actions are called output variables.

Actions outside the system are divided into two groups, control variables, which can be manipulated, and disturbance on which it is not possible any control.

Within systems is the concept of control system. A control system is a type of system that is characterized by the presence of a number of elements which allow to influence the operation of the system. The purpose of a control system is achieved, by manipulating the control variables, a domain on the output variables, so they reach predetermined values (slogan).

An ideal control system should be able to achieve its goal to satisfy the following requirements:

- Ensuring stability and, particularly, be robust against disturbances and errors in the models.
- Be as efficient as possible, according to a predetermined criterion. Usually this criterion is that the control action on the input variables is achievable, avoiding abrupt and unrealistic behavior.
- Be easily implemented and convenient to operate in real time using a computer.

The basic elements that are part of a control system and allow their manipulation are:

- Sensors. You can see the values of the variables of the system.
- Controller. Using the values measured by sensors and the slogan imposed calculated action to be applied to modify the control variables based on some strategy.
- Actuator. It is the mechanism that performs the action calculated by the controller and modifying the control variables.

3.3 Control Strategy

The control strategy refers to the nature and direction of the linkages between the variables measured and / or controlled and control variables.

There are two types of strategies depending on the nature of the information used to calculate the action of system control, open loop and closed loop.

Open loop: the control action is calculated knowing the dynamics of the system, the slogans and estimating the disturbances. This control strategy can compensate for delays inherent in the system in anticipation of user needs. However, the open loop is generally insufficient, due to model errors and errors in the estimation of disturbances. It is therefore common association of open-loop closed loop, so that the closed loop to compensate the errors generated by the open loop.

Closed Loop: The control action is calculated based on the measured error between the controlled variable and the desired setpoint. Perturbations, but are unknown are considered indirectly through its effects on output variables. This type of control strategy can be applied whatever the controlled variable. The vast majority of the control systems being developed today are closed loop.

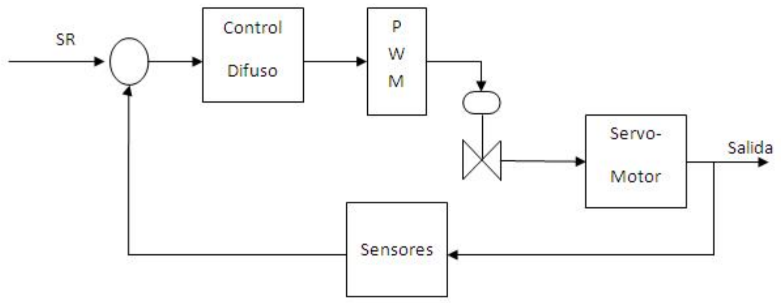

Fig. 3. General Scheme of a system based on Closed Loop.

3.4 Design Methods

The methods of systems analysis and design of controllers have evolved the same way that most of the tools used in engineering. The evolution in computer systems has enabled environments currently available that can perform dynamic simulations (eg Matlab). That is why in recent years, these advances have enabled research and applications in automatic control theory are gone from using an implementation monovariable analog and digital implementation of a multivariable.

In an analog scheme, all variables are functions of a continuous time, whereas in a digital pattern variables are known at some instant, in a discrete time.

Digital control systems have a number of advantages such as lower susceptibility to deterioration due to the passage of time or environmental factors, presents some components less sensitive to noise or vibration signals have greater flexibility when program, or have better sensitivity to parameter variation. In contrast, the evolution of computers and computing capabilities to reduce the disadvantages of digital controllers, so its use has recently spread in large amounts.

The variables measured in the system via the sensors reach the controller in analog form, so that in the case of a digital controller is necessary to use two signal converters, one analog-digital, which allows the signal to discretize the controller can perform necessary calculations, and a digital-analog conversion of the control commands calculated by the controller in continuous signals, so that the actuators can execute the necessary changes.

4 Description of Control Software Design

The development and implementation of the software was based on the needs that were presented, divided by modules, this in order to cope with each of the needs independently. The following describes each of the modules are the need and the solution suggested.

4.1 Concatenation of the PWM module

The PWM is the module that will allow manipulation of the direction of smart car, it needs to have a modulation rate of 20ms to excite the servomotor. The proposed code for this purpose is as follows;

```
#include < hidef.h >                          /* common defines and macros */
#include < mc9s12dt256b.h >                   /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12dt256b"
#define compPWM PWMDTY0                       // PWM Channel 0 Input 8-bit
//#define compPWN PWMDTY1                     // PWM Channel 1 Input 8-bit
//#define DDRA SPA                            // Output Port Selection

void main(void)          {

PWME_PWME0=1;       // Enabling the Pulse Width Channel 0
PWME_PWME0=1;       // Enabling the Pulse Width Channel 1
PWMPOL_PPOL0=1;     // Enabler of Polarity Pulse Width Channel 0
PWMPOL_PPOL1=1;     // Enabler of Polarity Pulse Width Channel 1
PWMCLK_PCLK0=1;     // Selecting the Clock Pulse Width Channel 0
PWMCLK_PCLK0=1;     // Selecting the Clock Pulse Width Channel 1
PWMCAE_CAE0=1;      // Centered Alignment Output Mode Channel 0
PWMCAE_CAE1=1;      // Centered Alignment Output Mode Channel 1
PWMSCLA_BIT0=1;     // Prescaled the clock for channel 0
PWMSCLA_BIT0=0;     // PWM Prescale Clock Select Register SA
PWMSCLA_BIT1=0;     // PWM Prescale Clock Select Register SA
PWMSCLA_BIT2=1;     // PWM Prescale Clock Select Register SA
PWMPER0=0x04;       // 9 Modulation period comes to 255
PWMPER1=0xFF;       // ff Modulation period comes to 255
PWMCTL_CON01=1;     // Concatenate channels 0 and 1
DDRP_DDRP0= 0x1;    // PWM0 OUTPUT AS PORT P0
}
```

Enabling PWM Channel 2 Motor Rear

```
PWME_PWME2=1;       // Pulse Width Channel 0 Enable
PWMPOL_PPOL2=1;     // Pulse Width Channel 0 Polarity
PWMCLK_PCLK2=1;     // Pulse Width Channel 0 Clock Select
PWMCAE_CAE2=1;      // Center Aligned Output Mode on channel 0
PWMSCLB_BIT0=1;     // Prescaled the clock for channel 0
PWMSCLB_BIT1=1;     // Prescaled the clock for channel 0
PWMSCLB_BIT2=0;     // Prescaled the clock for channel 0
PWMPER2=0xFF;       // PWM Channel Period 0 Register
DDRP_DDRP2= 0x1;    // Data Direction Port P Bit 0 for out and 1 for input
```

4.2 Modulo del ADC

The ADC is the module that will convert our system into a digital system, as it will develop the function of converting the signals from the sensors to '1 'or '0' depending on the case. Thus generating the PWM input data for it to perform its function. The corresponding code is as follows;

```
#include < hidef.h >                          /* common defines and macros */
#include < mc9s12dt256b.h >                   /* derivative information */
pragma LINK_INFO DERIVATIVE "mc9s12dt256b"
#define compPWM PWMDTY0                       // PWM Channel 0 Input 8-bit
```

```
void main(void) {

ATD0CTL2 = 0b10000011;
ATD0CTL3 = 0b10000000;
ATD0CTL4 = 0b10000000;
ATD0CTL5 = 0b10100101;
ATD0STAT0 = 0b0000000;
ATD0DIEN = 0b00100000;
}
```

4.3 Modulo de Logica Difusa

The fuzzy logic software module is a control key algorithm, which generates the error compensation. The following is the code that corresponds to this module.

```
#include hidef.h                                   /* common defines and macros */
#include mc9s12dt256.h                             /* derivative information */
#pragma LINK_INFO DERIVATIVE "mc9s12dt256b"
#define FuzzySets        7
#define FuzzySets_out    7
#define Entradas         2
#define Sep 0xFE         // El valor especial $FE se interpreta por REV como
#define End_rule 0xFF    //Marca el fin de las Reglas const enum {
```

Variables que almacenan la Fuzzyficación, " Etiquetas e, *d*e y *d*u"

Entrada Error Actual	Derivada del Error	Salida Delta U
e_G_negativo	*d*e_G_negativo	*d*u_G_negativa
e_M_negativo	*d*e_M_negativo	*d*u_M_negativa
e_P_negativo	*d*e_P_negativo	*d*u_P_negativa
e_Cero	*d*e_Cero	*d*u_Cero
e_P_positivo	*d*e_P_positivo	*d*u_P_positiva
e_M_positivo	*d*e_M_positivo	*d*u_M_positiva
e_G_positivo	*d*e_G_positivo	*d*u_G_positiva

List of Rules of Inference

```
unsigned char Fuzzy_IN_OUT[Entradas*FuzzySets];
unsigned char Fuzzy_out[FuzzySets_out];
const unsigned char Fuzzy_Reglas[ ]= {

    /*IF*/e_G_negativo,   /*AND*/de_G_negativo,   /*THEN*/Sep,du_G_negativa,Sep
    /*IF*/e_M_negativo,   /*AND*/de_G_negativo,   /*THEN*/Sep,du_G_negativa,Sep
    /*IF*/e_P_negativo,   /*AND*/de_G_negativo,   /*THEN*/Sep,du_G_negativa,Sep
    /*IF*/e_Cero,         /*AND*/de_G_negativo,   /*THEN*/Sep,du_G_negativa,Sep
    /*IF*/e_P_positivo,   /*AND*/de_G_negativo,   /*THEN*/Sep,du_M_negativa,Sep
    /*IF*/e_M_positivo,    /*AND*/de_G_negativo,    /*THEN*/Sep,du_P_negativa,Sep
    /*IF*/e_G_positivo,    /*AND*/de_G_negativo,    /*THEN*/Sep,du_Cero
    /*IF*/e_G_negativo,    /*AND*/de_M_negativo,    /*THEN*/Sep,du_G_negativa,Sep
    /*IF*/e_M_negativo,    /*AND*/de_M_negativo,    /*THEN*/Sep,du_G_negativa,Sep
    /*IF*/e_P_negativo,    /*AND*/de_M_negativo,    /*THEN*/Sep,du_G_negativa,Sep
    /*IF*/e_Cero,          /*AND*/de_M_negativo,    /*THEN*/Sep,du_M_negativa,Sep
    /*IF*/e_P_positivo,    /*AND*/de_M_negativo,    /*THEN*/Sep,du_P_negativa,Sep
    /*IF*/e_M_positivo,    /*AND*/de_M_negativo,    /*THEN*/Sep,du_Cero,Sep
    /*IF*/e_G_positivo,    /*AND*/de_M_negativo,    /*THEN*/Sep,du_P_positiva,Sep
    /*IF*/e_G_negativo,    /*AND*/de_P_negativo,    /*THEN*/Sep,du_G_negativa,Sep
    /*IF*/e_M_negativo,    /*AND*/de_P_negativo,    /*THEN*/Sep,du_G_negativa,Sep
    /*IF*/e_P_negativo,    /*AND*/de_P_negativo,    /*THEN*/Sep,du_M_negativa,Sep
    /*IF*/e_Cero,          /*AND*/de_P_negativo,    /*THEN*/Sep,du_P_negativa,Sep
    /*IF*/e_P_positivo,    /*AND*/de_P_negativo,    /*THEN*/Sep,du_Cero, Sep
    /*IF*/e_M_positivo,    /*AND*/de_P_negativo,    /*THEN*/Sep,du_P_positiva,Sep
    /*IF*/e_G_positivo,    /*AND*/de_P_negativo,    /*THEN*/Sep,du_M_positiva,Sep
    /*IF*/e_G_negativo,    /*AND*/de_Cero,          /*THEN*/Sep,du_G_negativa,Sep
    /*IF*/e_M_negativo,    /*AND*/de_Cero,          /*THEN*/Sep,du_M_negativa,Sep
    /*IF*/e_P_negativo,    /*AND*/de_Cero,          /*THEN*/Sep,du_P_negativa,Sep
    /*IF*/e_Cero,          /*AND*/de_Cero,          /*THEN*/Sep,du_Cero,Sep
    /*IF*/e_P_positivo,    /*AND*/de_Cero,          /*THEN*/Sep,du_P_positiva,Sep
    /*IF*/e_M_positivo,    /*AND*/de_Cero,          /*THEN*/Sep,du_M_positiva,Sep
    /*IF*/e_G_positivo,    /*AND*/de_Cero,          /*THEN*/Sep,du_G_positiva,Sep
    /*IF*/e_G_negativo,    /*AND*/de_P_positivo,    /*THEN*/Sep,du_M_negativa,Sep
    /*IF*/e_M_negativo,    /*AND*/de_P_positivo,    /*THEN*/Sep,du_P_negativa,Sep
    /*IF*/e_P_negativo,    /*AND*/de_P_positivo,    /*THEN*/Sep,du_Cero,Sep
    /*IF*/e_Cero,          /*AND*/de_P_positivo,    /*THEN*/Sep,du_P_positiva,Sep
    /*IF*/e_P_positivo,    /*AND*/de_P_positivo,    /*THEN*/Sep,du_M_positiva,Sep
    /*IF*/e_M_positivo,    /*AND*/de_P_positivo,    /*THEN*/Sep,du_G_positiva,Sep
    /*IF*/e_G_positivo,    /*AND*/de_P_positivo,    /*THEN*/Sep,du_G_positiva,Sep
    /*IF*/e_G_negativo,    /*AND*/de_M_positivo,    /*THEN*/Sep,du_P_negativa,Sep
    /*IF*/e_M_negativo,    /*AND*/de_M_positivo,    /*THEN*/Sep,du_Cero,Sep
    /*IF*/e_P_negativo,    /*AND*/de_M_positivo,    /*THEN*/Sep,du_P_positiva,Sep
    /*IF*/e_Cero,          /*AND*/de_M_positivo,    /*THEN*/Sep,du_M_positiva,Sep
    /*IF*/e_P_positivo,    /*AND*/de_M_positivo,    /*THEN*/Sep,du_G_positiva,Sep
    /*IF*/e_M_positivo,    /*AND*/de_M_positivo,    /*THEN*/Sep,du_G_positiva,Sep
    /*IF*/e_G_positivo,    /*AND*/de_M_positivo,    /*THEN*/Sep,du_G_positiva,Sep
    /*IF*/e_G_negativo,    /*AND*/de_G_positivo,    /*THEN*/Sep,du_Cero,Sep
    /*IF*/e_M_negativo,    /*AND*/de_G_positivo,    /*THEN*/Sep,du_P_positiva,Sep
    /*IF*/e_P_negativo,    /*AND*/de_G_positivo,    /*THEN*/Sep,du_M_positiva,Sep
    /*IF*/e_Cero,          /*AND*/de_G_positivo,    /*THEN*/Sep,du_G_positiva,Sep
    /*IF*/e_P_positivo,    /*AND*/de_G_positivo,    /*THEN*/Sep,du_G_positiva,Sep
    /*IF*/e_M_positivo,    /*AND*/de_G_positivo,    /*THEN*/Sep,du_G_positiva,Sep
    /*IF*/e_G_positivo,    /*AND*/de_G_positivo,    /*THEN*/Sep,du_G_positiva,Sep
   End_rule
};
```

Membership Functions

```
const unsigned char Membership_Functions[ ]=
```

Trapezoidal function

	Error	Actual			Error	Derivado	
0x00,	0x26,	0x00,	0x07	0x00,	0x26,	0x00,	0x07,
0x04,	0x51,	0x07,	0x07,	0x04,	0x51,	0x07,	0x07,
0x2F,	0x7B,	0x07,	0x07,	0x2F,	0x7B,	0x07,	0x07,
0x58,	0xA5,	0x06,	0x06,	0x58,	0xA5,	0x06,	0x06,
0x84,	0xD0,	0x07,	0x07,	0x84,	0xD0,	0x07,	0x07,
0xAE,	0xFB,	0x07,	0x07,	0xAE,	0xFB,	0x07,	0x07,
0xD9,	0xFF,	0x07,	0x00,	0xD9,	0xFF,	0x07,	0x00,

```
 };
```

FM Singleton para la salida Delta U

```
const unsigned char Fuzzy_out_FM[ ]= {

 0x03    /*du_Grande_negativa*/
 0x2B    /*du_Mediana_negativa*/
 0x55    /*du_Pequeña_negativa */
 0x80    /*du_Cero*/
 0xAA    /*du_Pequeña_positiva*/
 0xD5    /*du_Mediana_positiva*/
 0xFF    /*du_Grande_positiva */

   ;
unsigned char e=92, de=20, du=0;

void Fuzzy_function(void);

void main(void) {
```

ADC Initialization Module

PWM Initialization Module

```
EnableInterrupts;
DDRB=0XFF;
Fuzzy_function();
for(;;) {} /* wait forever */
}
```

Kernel fuzzy inference

```
void Fuzzy_function(void){
__asm
{
```

```
Fuzzy:
  ldx   #Funciones_de_Membresia   // ;Apunta a definiciones de FM
  ldy   #Fuzzy_IN_OUT             // Apunta a tabla de entradas fuzzyficadas
 ldaa   e                         // Carga el valor del Error Actual en A
 ldab   #FuzzySets                // 7 FM para la entrada Error Actual
Fuz_loop:
 mem                  //Evalúa una FM hallando el valor y para el Error
 dbne   B,Fuz_loop    //Lazo para evaluar las 7 FM del Error Actual
  ldaa  de            //Carga el valor de la Derivada del Error en A
  ldab  #FuzzySets    //7 FM para la entrada Derivada del Error
Fuz_loop1:
 mem                  // Evalúa una FM
 dbne   B,Fuz_loop1   // Lazo para evaluar las 5 FM de la Derivada del Error
  ldab  #FuzzySets    // contador del lazo
Evalua_regla:
   clr  1,Y+               // Limpia sal. Fuzzyficada e incr. Apunt.
 dbne   B,Evalua_regla     // Lazo para limpiar todas las sal. Fuzzyficadas
  ldy   #Fuzzy_IN_OUT      //Apunta a ent y sal. fuzzyficadas
  ldx   #Fuzzy_Reglas      //Apunta al 1er elemento de Reglas
 ldaa   #$FF               //Inicia A (y limpia bit V)
  rev                      //Procesa lista de Reglas
Defuzzy:
  ldy   #Fuzzy_out        // Apunta a salidas fuzzyficadas
  ldx   #Fuzzy_out_FM     // Apunta a posiciones singleton
 ldab   #FuzzySets_out    // 7 salidas fuzzy por salida crisp
 wav                      // Hace suma por prom ponderado
 ediv                     // Divide por suma ponderada
  tfr   Y,D               // Pone resultado en A:B
 stab   du                // Almacena salida del sistema
   }
   }
```

5 Description of Key Technical Parameters of Car Models

The main descriptions of the parameters of the smart car are as follows;

5.1 Total Weight and Dimensions (Length and Width) of the Reengineered Car Model

The main description of the parameters of the dimensions of the smart car race are specified by the rules of competition, so here are the parameters in Table I, where we can appreciate the reengineered Car Model parameters with the assembly of all components.

Table I. Parameters to Car Model

	Total Weight	Dimensions
Load	870gr	31.2cm X 17cm x 11cm

5.2 Power Consumption and Total Capacitance of all Capacitors

- Voltage ENTRY 7.1 Volts
- Output voltage setting 5 Volts
- Power generated by the batteries 7.12 Volts
- Amps Current consumption 0.43
- Power consumed by the engine 3.05 Watts

- Sensor 260mA
- Microcontroller 50mA
- Power module 10mA
- Servomotor 110mA
- Motor 980mA

5.3 Count and Type of Sensors Used

The line follower consists of basic electrical systems and sensors for control. We carried out a control system to build the follower system, governed by the HC12 microcontroller that is capable of following the black line inside a white track or platform.

The sensor has been used for the development of this function has been the CNY70, which is a reflective infrared sensor type.

To form the sensor signals CNY70 entry have enabled microcontroller ADC module.

- When a sensor detects the white background, the line input of ATD_PORT it is connected goes to 0.
- When a sensor is on the black line, the line input of ATD_PORT it is connected comes a 1.
- The entrance to the servomotor is connected to Microcontroller's PWM output.
- The output of the servomotor is connected to the vehicle steering system.

The numbers of sensors used in the completed form to monitor the line are 8 sensors CNY70 with a separation of 2.5cm apart.

5.4 Number of Servo Motors Besides the Existing Driving Motors and Rudder Motors of the Car Model

The amount of servo motor used in the smart car is 1 servo motor, which is used as the handle of the direction of it, based on the received signal of the PWM.

As is only used a rudder motors in the smart car for the drive of the same engine friction. The characteristics and main parameters of the engine are shown in Tables II and III.

Table II. Motor Reference Guide

Model	RS380-ST
Winding	11480
Voltage	24.0V

Table III. Parameters to Motor Reference Guide

No Load		Max Efficiency			Max Output			Stall	
Current	Speed	Current	Speed	Torque	Current	Speed	Torque	Current	Speed
A	rpm	A	rpm	g.cm	A	rpm	g.cm	A	rpm
0.036	4750	0.16	3890	60	0.38	2375	166	0.73	332