Chapter 6

Remotely Train Control with the Aid of PIC32 Microcontroller

Mostefa Ghassoul

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/57365

1. Introduction

Automatic train control is very difficult due to two main reasons. One is the metrological conditions such as wind, rain, snow, heat, tear and wear which affects its behavior and secondly the changing load whether that load is passengers or goods, where the number of passengers may change from station to station, or change in goods as well. This renders the system to be highly non linear. So a non linear scheme is required. One feasible technique is by combining a PI with a fuzzy controller. To make the controller efficient, a 32 bits microchip PIC32 microcontroller is used. Programming PIC32 using fuzzy is very tedious, so it would be far better to use the MATLAB SIMULINK fuzzy block set. To program the controller using SIMULINK, special block sets are used called MICROCHIP block sets. Those have been developed by somebody called Kerhuel (1) and are offered for demo purpose with limited inputs. The chapter is divided into five sections. After this introduction, Section2 addresses the speed capture using a PIC18F452 mounted on the train, as well as the data transmission and data reception by the PIC32 level. Section3 discusses the train modeling. Section4 discusses the PI fuzzy logic controller and its implementation in SIMULINK. Section5 addresses the testing of the system, first by simulating the train, then two station test and three station test. Section6 presents the conclusion of the work presented here. The chapter is concluded by looking at possible alternatives to SIMULINK such as NI LABVIEW and Inform Fuzzy TECH. Last but not the least, a list of references is presented at the end of the chapter. A block diagram of the scheme is shown in figure (1).

2. Train speed capture

Before any control takes place, the train speed has to be captured. This is done by using a light source (LED), which emits narrow rays towards a reflector mounted on one of the train wheels. The reflected light is picked up by a phototransistor. The output of the phototransistor is in the range of 1.62-2.07 V on no light, and 2.31- 2.89V on reflected light presence. This voltage is fed to a buffer (LM358N) for matching purpose. The buffer output is compared with a 2.25V voltage through another LM358N operational amplifier, to produce a pulse train, with zero volt for low voltage (<2.25V) and 5 V for high voltage (>2.25V). The duration of the pulse is proportional to the train speed. The speed capture circuit is shown in figure (2).

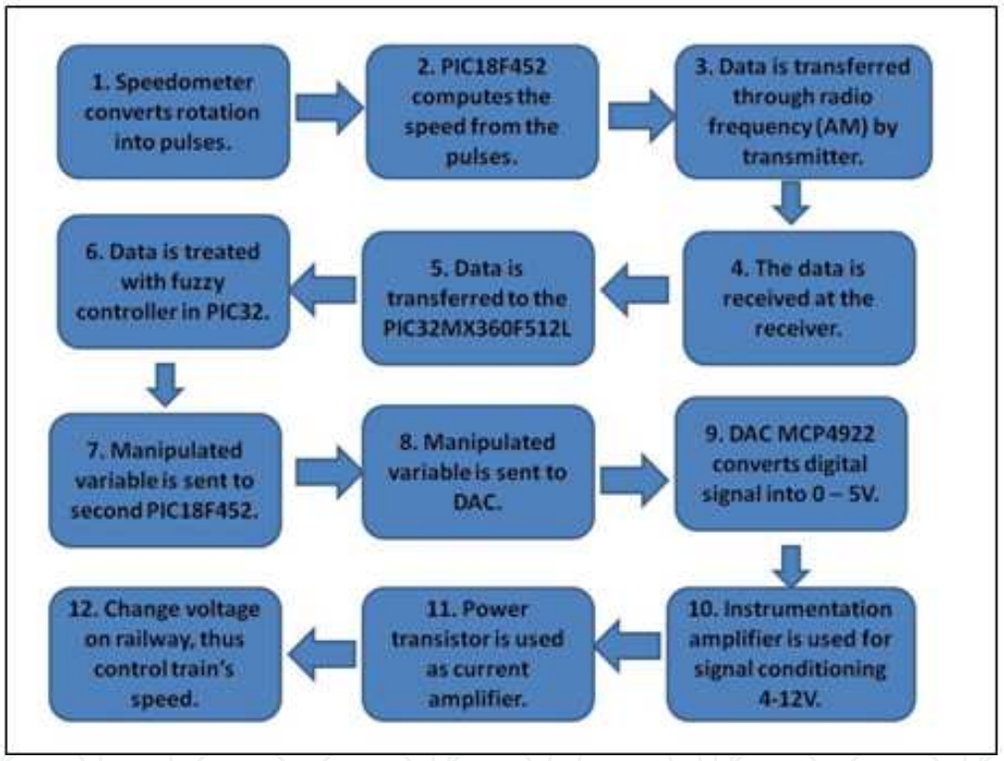

Figure 1. Block diagram of the system

The pulses are captured through the data capture port of the microchip PIC18F452 microcontroller, where each two successive leading edges are detected. The PIC18F452 has four 16 bit timers. In this project only Timer0 is used and programmed as a free running timer. On each leading edge, the timer value is read in a buffer. Then its value is read on the following leading edge into a second buffer. The difference between the two values gives the duration of the pulse. The shorter is the pulse, the faster is the train. So the speed is inversely proportional to the pulse duration.

V = K/T where V is the speed, T is the pulse period and K is the proportionality constant. It was found that the value of k is equal to 0.14192 V.S.

http://dx.doi.org/10.5772/57365

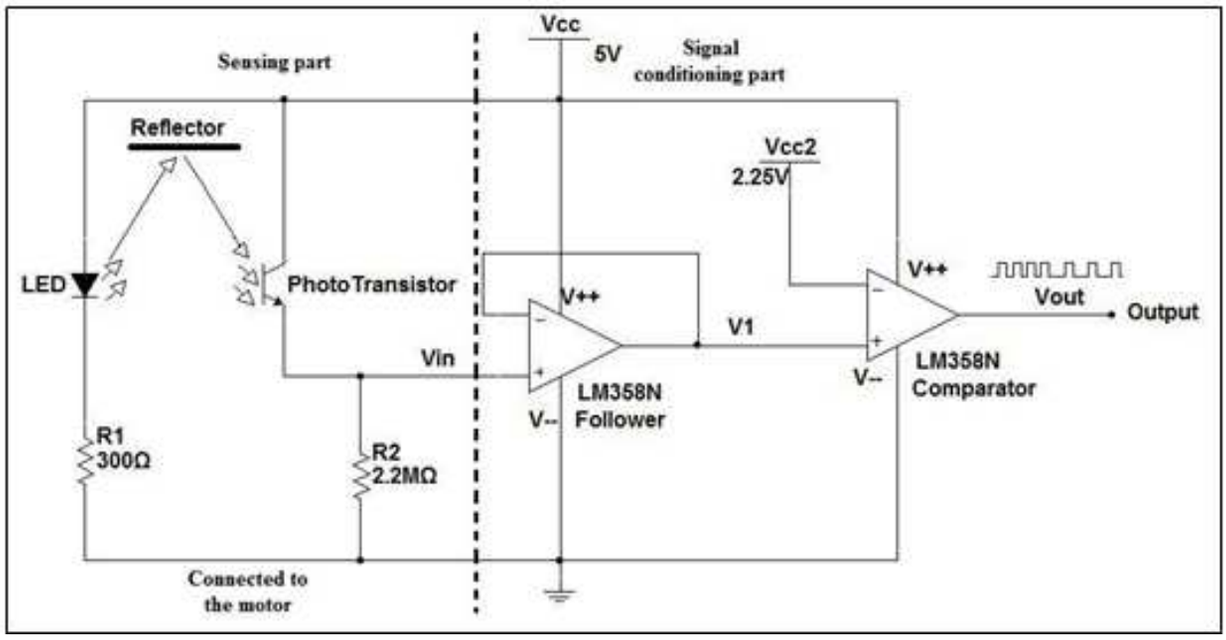

Figure 2. Train speed capture circuit

2.1. Speed transmission

The speed capture circuit is mounted on the train. So once the speed is computed, The signal is sent from the microcontroller using the Universal Synchronous Asynchronous Receiver Transmitter serial port (USART). it is then modulated using amplitude shift keying (ASK) in order to be radio transmitted. The modulator is of type SHY-J6122TR transmitter/receiver. It is capable of transmitting radio frequencies in the band 300-450MHz. In the control room, is the main controller which is nothing less than the powerful 32 bit PIC32 microcontroller. The choice of this device is for several reasons. 1- It has a 512 Kbytes of flash memory; so it could accommodate any program of this type of application. 2- It has a large stack, so it could accommodate all the if statements, produced by SIMULINK fuzzy controller algorithm, no matter how many rules are implemented without jumping the stack. 3- No truncation is needed because of the size of mantissa which is 32 bits. 4- The availability of the SIMILINK block sets, which makes the programming of the controller very easy. 5- up to 32 kbytes of RAM, for storing variables and tables when required. The speed signal is read into the controller through the USART, after being received and demodulated using a second SHY-J6122TR transmitter/receiver. The transmitting microcontroller was programmed using PCC C language, and the subroutine is shown below:

Result1 and result2 are the current and previous timer readings, corresponding to edge capture of the current and previous edge. Line 18 defines the configuration of the serial port (USART). Line 26 configures the data capture1, where data is captured on every rising edge. Line 29 configures TIMER 1 as a 16 bit timer, with 1:1 prescaler, and timer source is capture1 (CCP), and the configuration circuit is shown in figure (3).

```
#include <p18f452.h>
#include <capture.h>
#include <timers.h>
#include <stdlib.h>
#include <usart.h>

#pragma config WDT = OFF

unsigned int result1;
unsigned int result2;
unsigned int delta;
unsigned int speed;

void main (void)
{

        //configure USART
        OpenUSART ( USART_TX_INT_OFF & USART_RX_INT_OFF &
        USART_ASYNCH_MODE & USART_EIGHT_BIT & USART_CONT_RX
&
        USART_BRGH_HIGH, 207 );

        //configure PORTC, bit3 must be input RC2, bit6 must be output RC6
        TRISC = 0x04;

        // Configure Capture1
        OpenCapture1 ( C1_EVERY_RISE_EDGE & CAPTURE_INT_OFF );

        // Configure Timer1, internal clock is used which means Fosc/4
        OpenTimer1 ( TIMER_INT_OFF & T1_16BIT_RW & T1_SOURCE_INT
&
                T1_PS_1_1 & T1_OSC1EN_OFF & T1_SOURCE_CCP );

        while (1)
        {

                while (!PIR1bits.CCP1IF); // Wait for event
                result1 = ReadCapture1(); // read result
                PIR1bits.CCP1IF = 0;//clear

                while (!PIR1bits.CCP1IF);
                result2 = ReadCapture1();
                PIR1bits.CCP1IF = 0;

                delta = result2 - result1;

                speed = 141952/delta;

                WriteUSART ( speed );
        }
}
```

http://dx.doi.org/10.5772/57365

2.2. Speed reception by the main controller

Once the speed is air transmitted, it is received into the main controller which is no more than the microchip PIC32MX360F512L microcontroller. The choice of this controller is because of the availability of the SIMILINK block sets to drive it. At start, the microcontroller has to be configured.

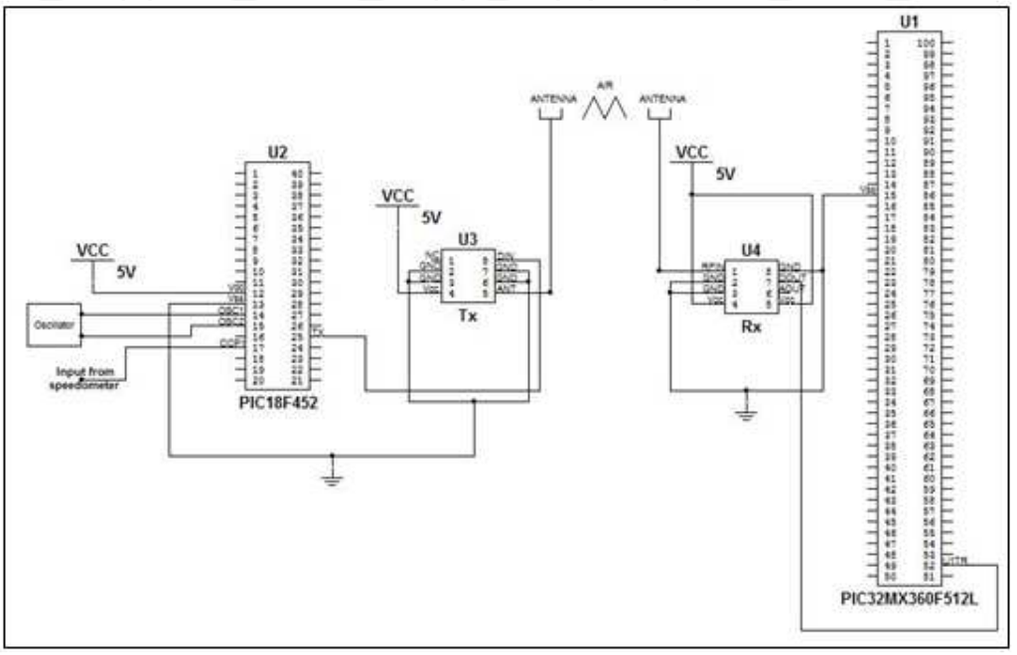

Figure 3. Circuit showing the transmitter and the receiver of the speed

This is done using three SIMULINK block sets. The first one is called "master". By clicking its icon, a windows opens where one configures the PIC, by selecting the type of controller, the port used, the clock which UART is used. Then UART1 icon (two UARTs are available on the chip) is clicked so that UART1 configuration pops up so it could be configured by setting the baud rate etc. Figure (4) shows the block set configuration.

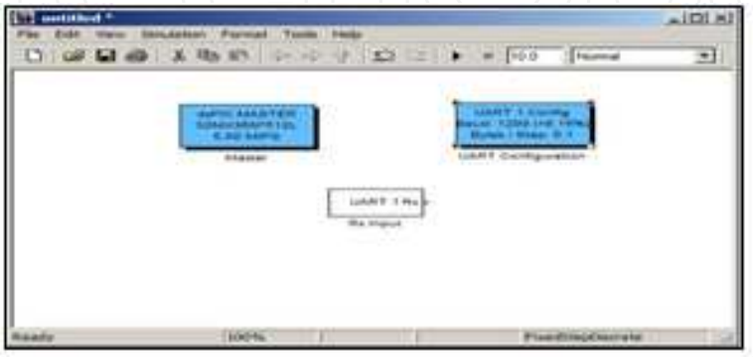

Figure 4. PIC32 port configuration using SIMULINK

3.Train model

Before any control to be applied, the train was first modeled using MATLAB system identification tool (SIT). This is done by entering a series of input voltages together with corresponding speeds. Table (1) shows the input voltage and the corresponding train speed. After opening SIT, "Time Domain Data" was selected to write workspace variables (voltage and speed) as shown in figure (5). After importing the data and estimating the model to be second order system with a time delay which represents the model for a DC motor, process model parameters were found. This is shown in figure (6).

	>> voltage=v
speed =	voltage =
1.7100	3.9900
3.4700	5.0000
5.1900	6.0400
6.3700	6.9900
7.8700	7.9900
9.5900	9.0000
11.2900	10.0100
13.2100	11.0000
14.8900	11.9900

Table 1. Speed (cm/s) as a function of input voltage (V)

4. Fuzzy logic controller

In any standard book on fuzzy control, fuzzy logic control is defined to be a practical alternative for a variety of challenging control applications since it provides a convenient method for constructing non-linear controllers via the use of heuristic information. Since heuristic information may come from an operator who has acted as “a human in the loop” controller for a process. In the fuzzy control design methodology, a set of rules on how to control the

http://dx.doi.org/10.5772/57365

Figure 5. Importing data from workspace

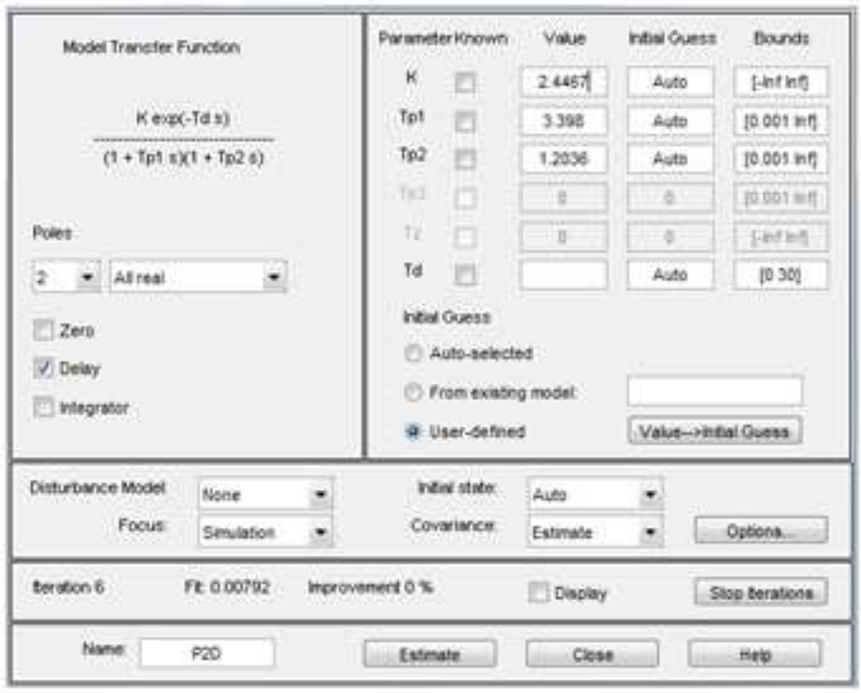

Figure 6. The modeling parameter of the train

process is written down and then it is incorporated into a fuzzy controller that emulates the decision making process of the human. In other cases, the heuristic information may come from a control engineer who has performed extensive mathematical modelling, analysis and development of control algorithms for a particular process. The ultimate objective of using fuzzy control is to provide a user-friendly formalism for representing and implementing the ideas we have about how to achieve high performance control. Apart from being a heavily used technology these days, fuzzy logic control is simple, effective and efficient(2). In this section, the structure, working and design of a fuzzy controller is discussed in detail through an in-depth analysis of the development and functioning of a fuzzy logic speed controller.

The general block diagram of a fuzzy controller is shown in figure (7). The controller is composed of four elements:

- A Rule Base
- An Inference Mechanism
- A Fuzzification Interface
- A Defuzzification Interface

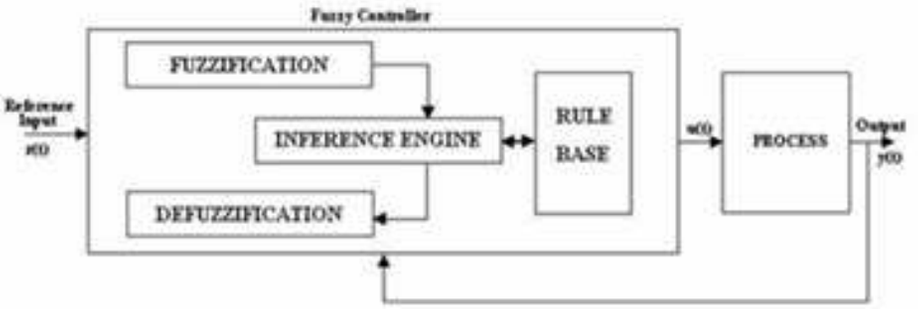

Figure 7. Fuzzy controler model

4.1. Rule base

This is a set of "If ……..then….." rules which contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control.

4.2. Inference mechanism

This emulates the expert's decision making in interpreting and applying knowledge about how best to control the plant.

4.3. Fuzzification interface

This converts controller inputs into information that the inference mechanism can easily use to activate and apply rules.

http://dx.doi.org/10.5772/57365

4.4. Defuzzification interface

It converts controller inputs into information that the inference mechanism into actual inputs for the process.

4.5. Selection of inputs and outputs

It should be made sure that the controller will have the proper information available to be able to make good decisions and have proper control inputs to be able to steer the system in the directions needed to be able to achieve high-performance operation.

The fuzzy controller is to be designed to automate how a human expert who is successful at this task would control the system. Such a fuzzy controller can be successfully developed using high-level languages like C, Fortran, etc. Packages like MATLAB® also support Fuzzy Logic.

4.6. Fuzzy sets and membership function

Given a linguistic variable Ui with a linguistic value Aij and membership function μ Aij(Ui) that maps Ui to [0, 1], a 'fuzzy set is defined as

Aij= {(Ui, μ Aij (Ui)); Ui ε υi}

The above written concept can be clearly understood by going through the following example. Suppose we assign Ui="VOLTAGE" and linguistic value A11="base", then A11 is a fuzzy set whose membership function describes the degree of certainty that the numeric value of the temperature, Ui ε υi, possesses the property characterized by A11. This is made even clearer by the fig (8).

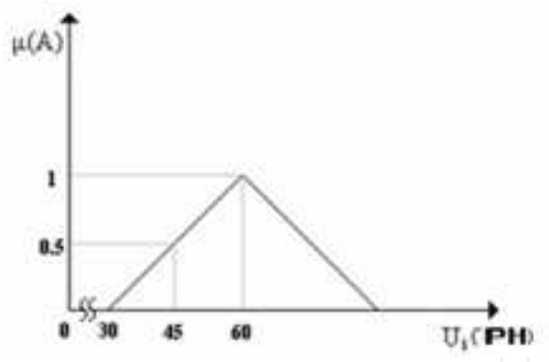

Figure 8. Triangular membership function

In the above example, the membership function chosen is triangular. There are some other membership functions lime Gaussian, Trapezoidal, Sharp peak, Skewed triangle, etc. Depending on the application and choice of the designer, the required one can be chosen (figure(9)).

It is well known that the train load is highly nonlinear due to the fact that the number of passengers keeps changing, as well as the goods, adding to that the unpredictable climatic conditions.. This makes the application of linear control techniques very difficult to implement.

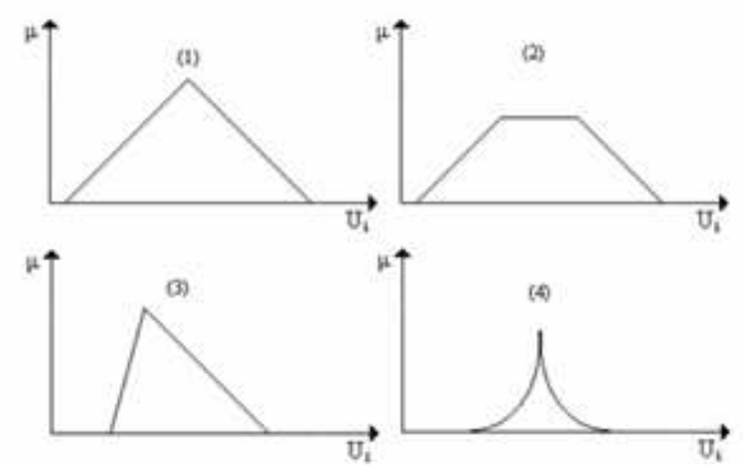

Figure 9. Triangular, 2) Trapezoid, 3) Skewed triangular, 4) Sharp peak

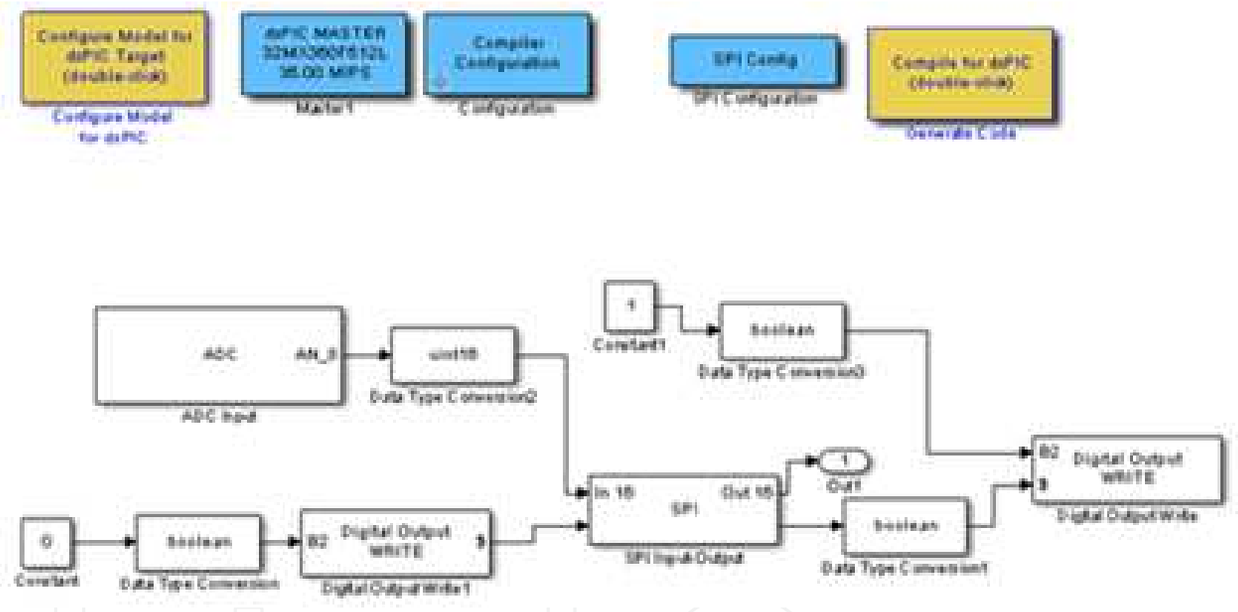

Figure 10. Different PIC32 block sets inserted into a SIMULINK model

So a feasible solution to make call to nonlinear techniques. A very suitable one for this type of application One is the fuzzy logic controller. On top of being simple to use because no transfer function is required, it could solve the nonlinearity problem if designed and tuned properly. To implement a fuzzy controller, using microcontroller, is a very tedious, time consuming and certainly not optimized whether it is developed using C language or assembly language. A better way is to make use of the optimized SIMULINK fuzzy controller block sets. But to do this, the PIC32 microcontroller resources have to be interfaced to SIMULINK. Luckily, this has been developed by a French researcher (1) and made available as a demo version with limited inputs/outputs. Those block sets are click and drag type, where they could be easily placed in

http://dx.doi.org/10.5772/57365

a SIMULINK pane and configured. Figure (10) shows the different PIC32 block sets inserted into a SIMULINK model to control the train.. Once the model is compiled, they are two ways on how to download it into the microcontroller. After compiling the model using MATLAB real time workshop (RTW), if successful, it will produce two files, one with extension exe and one with extension mcp. The executable (exe) file could be downloaded directly into the microcontroller using a boot loader and executed. But unfortunately there is no way it could be debugged. A better solution is to use the mcp file, by downloading into MPLAB by clicking on outside the MATLAB (refer to figure (11). This will open the project within MPLAB, with all the required files and headers. It is then rebuilt and downloaded it into the controller and executed. The advantage of doing so is that the project could be debugged on line using MPLAB facilities and if necessary, it could be modified. Figure (12) shows MPLAB project window, with all the required files.

Figure 11. Exporting file from MATLAB to MPLAB

Figure 12. Train project with all the required files and headers produced by SIMULINK

4.7. Fuzzy control implementation

In the case in hand (train control), the constructed fuzzy controller has two inputs which are the distance and the speed and one output which is the voltage which drives the train. Before using the fuzzy block set in SIMULINK, we have first to define the rules and conditions, by importing the fuzzy software into MATLAB environment. Then define the memberships, rules and conditions. Once that is done, the file has to be exported to MATLAB workspace so it will be recognized by the SIMULINK fuzzy logic block set call.

4.7.1. Fuzzy memberships and rules

The fuzzy controller is shown in figure (13). The model has two inputs, distance and speed and one output is the voltage.

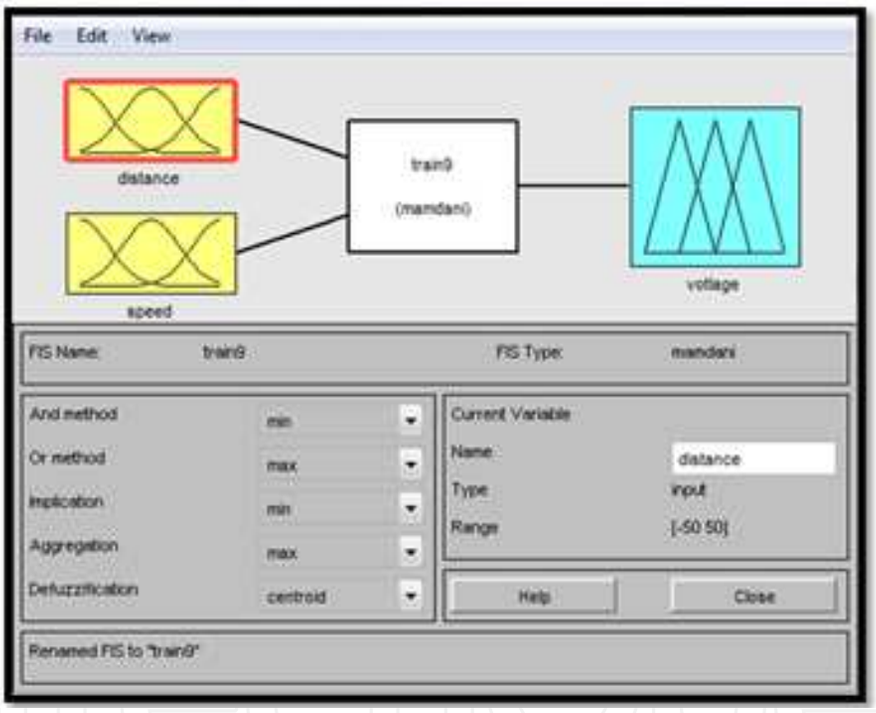

Figure 13. Train fuzzy model

The memberships were set. It is worth mentioning here that the train only accelerates at the start, then it cruises to its cruising speed and keeps the speed constant until it reaches the proximity of the breaking distance, it decelerates until it comes to a standstill. Figure (14) and figure (15) show the memberships of the inputs and output respectively. The memberships are of triangular type for accelerating and decelerating and trapezoidal shape for cruising. Using MATLAB software we design the rules and the functions we need for controlling the speed of the train. Figure (16) shows the fuzzy rules at the beginning, the train accelerates until it reaches cruising speed.

http://dx.doi.org/10.5772/57365

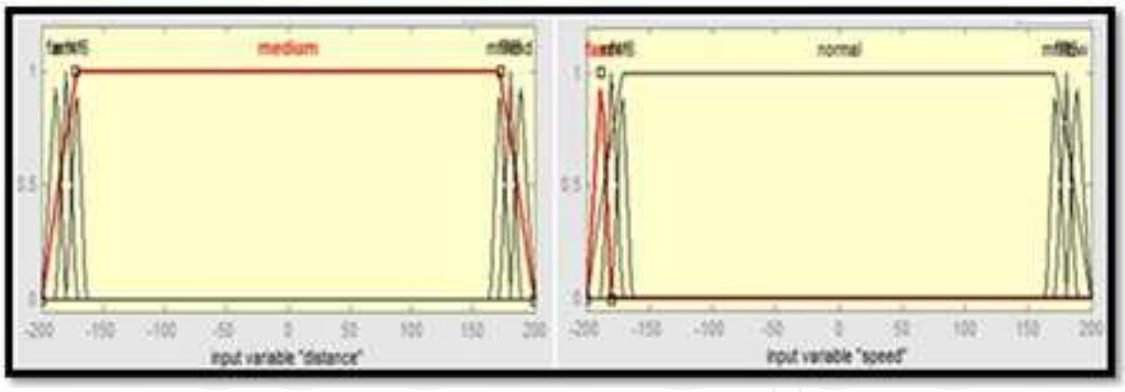

Figure 14. Membership of the inputs (distance and speed)

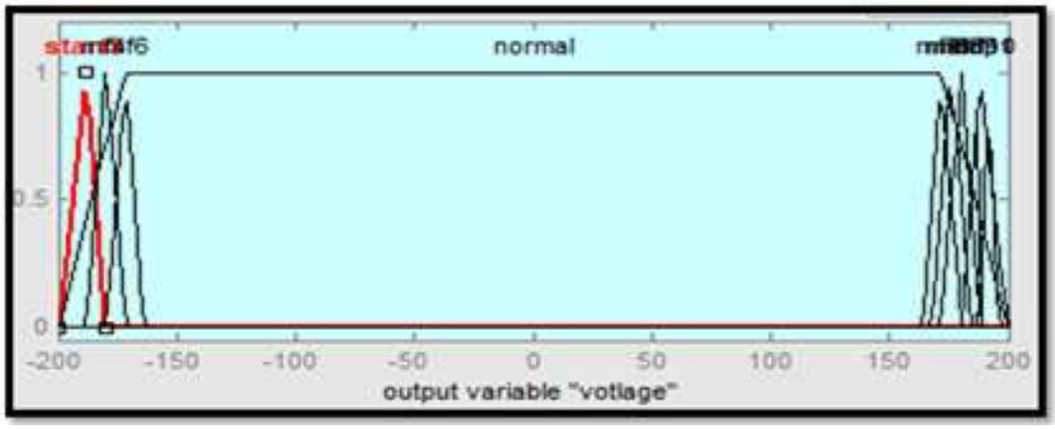

Figure 15. Membership of the output (voltage)

1. If (distance is far) and (speed is fast) then (votlage is start3) (1)
2. If (distance is medium) and (speed is normal) then (votlage is normal) (1)
3. If (distance is end2) and (speed is low2) then (votlage is stop2) (1)
4. If (distance is far1) and (speed is fast1) then (votlage is start1) (1)
5. If (distance is end1) and (speed is low1) then (votlage is stop1) (1)
6. If (distance is far2) and (speed is fast2) then (votlage is start2) (1)
7. If (distance is end) and (speed is low) then (votlage is stop) (1)
8. If (distance is medium) and (speed is normal) then (votlage is stop2) (1)
9. If (distance is medium) and (speed is normal) then (votlage is stop1) (1)
10. If (distance is medium) and (speed is normal) then (votlage is stop) (1)

Figure 16. Fuzzy rules for train control

For controlling the train, at the beginning the memberships are tuned so that the train cruises to normal speed (-200 _ -160 range). Once this is achieved, it stays there no mater how long is the trajectory, until it comes to the proximity of the stop station (range 160 _ 200), it decelerates until it comes to standstill following the stopping membership. Once the control action is computed, a deffuzification process takes over to generate the crisp output. Figure (17) shows the deffuzification.

5. Testing the system

Before the system was tested, the train was first modeled and simulated.

5.1. Simulating the system

Before the model was tested on the train, it was first simulated using the transfer function obtained in train model (see figure (6)), The SIMULINK model used is shown in figure (18). The control strategy used is a discrete proportional and integral action applied to the fuzzy controller. It is worth mentioning that the fuzzy controller is called by clicking on its action, and entering the name of the controller developed in MATLAB, it calls the model which was already exported into the workspace.. (Remember once model is developed in MATLAB, it is exported to the workspace).

The response of the model is shown in figure (19). Because the model used is a continuous one, the response is also continuous. This could be noticed in the transfer function and the PI controller where the S domain is used instead of the discrete sample. This is confirmed by the smoothness of the response. This will not be the case when online control of the train is implemented. It also shows the smoothness of the response whether at the start or stop phases.

5.2. Testing the controller using real data

The testing was carried out on a small train where the speed could be varied from zero to 31cm/s by varying the input voltage from zero to 12 V, first using only two stations then three stations.

5.2.1. Two station test

The model used for the test is shown in figure (20). The controller is reading the speed UART1 receiver(UART1 RX) of the PIC32, which was radio transmitted from the train. This feeds the PI controller. This time, a discrete digital integrator is used. This because the MATLAB real time workshop only complies PIC32 block sets using discrete form. It is worth mentioning that no model is present because a real train is used. The control variable is outputted using the UART1 transmitter (TX UART1) (how this signal is controlling the train will be discussed later). The train response is shown in figure (21). Figure (21) shows the smoothness by which the train accelerates, then cruise at a constant speed, then decelerates and stop exactly at the right stop.

http://dx.doi.org/10.5772/57365

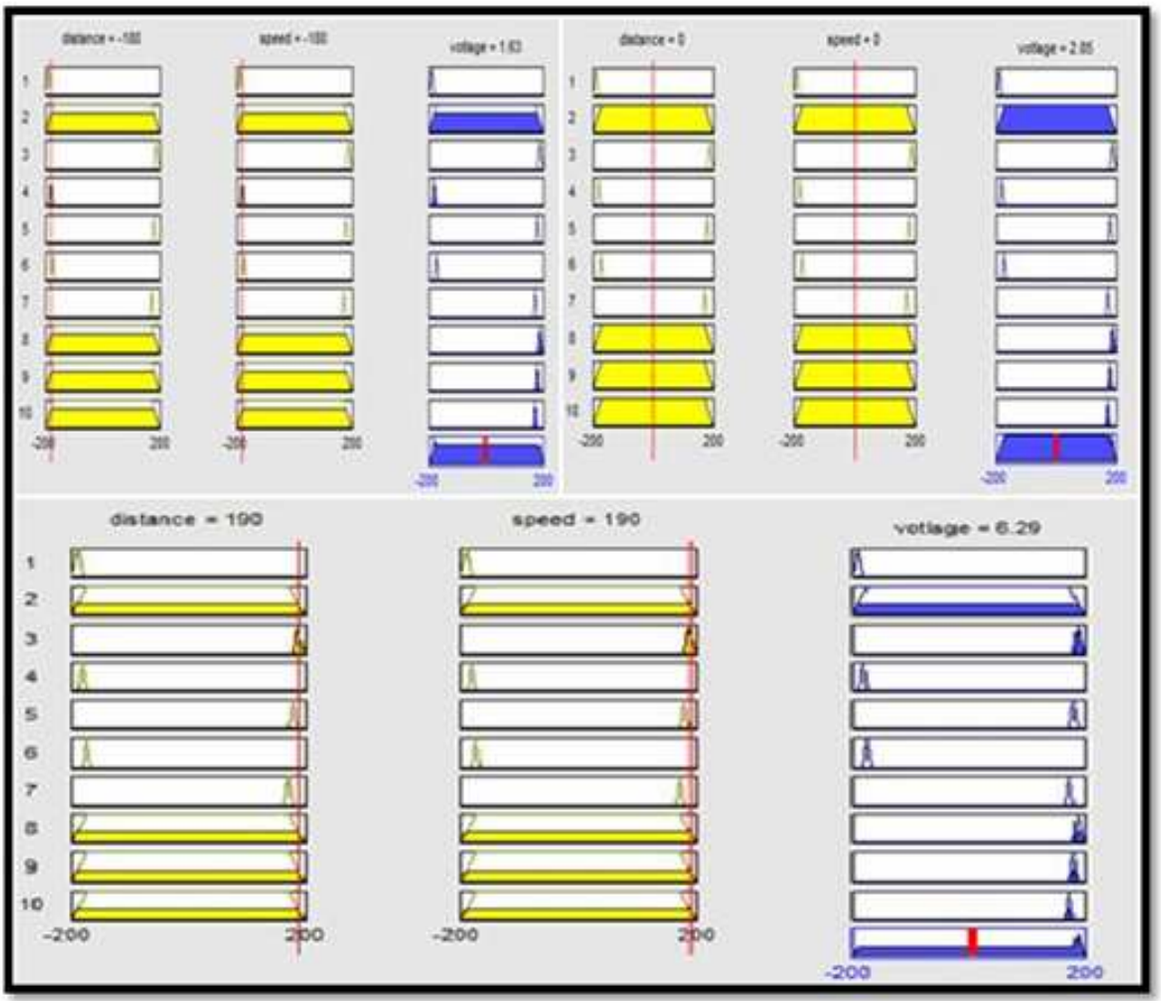

Figure 17. Deffuzification process

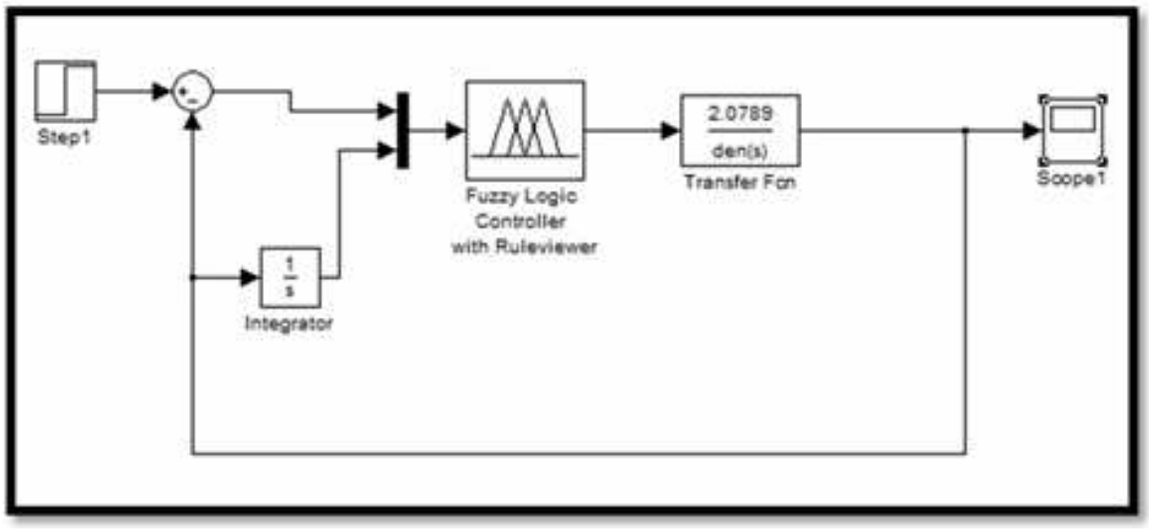

Figure 18. Simulation of the SIMULINK of the train model

Figure 19. SIMULINK simulation of the train model

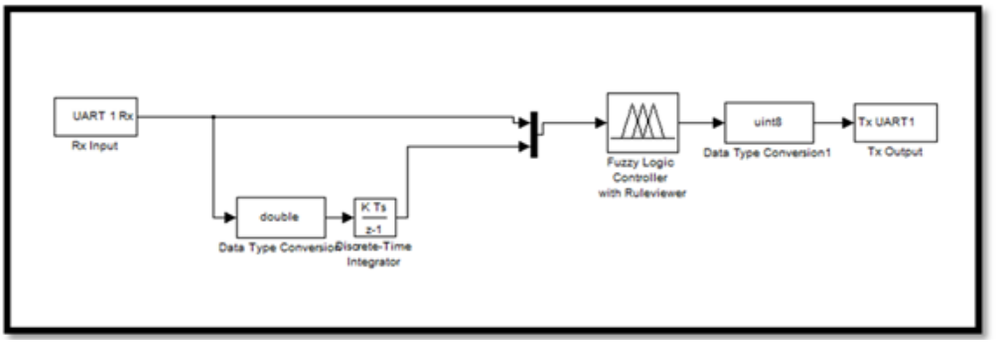

Figure 20. Train control between two stations

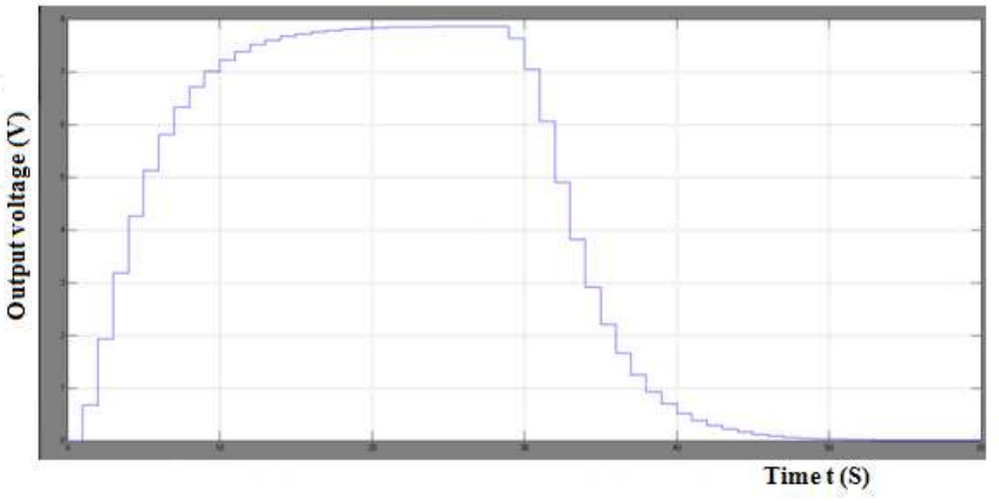

Figure 21. Train trajectory between two stations

http://dx.doi.org/10.5772/57365

5.2.2. Three station test

The train was then tested using three stations. Initially, it leaves station1 and accelerates, cruises, then decelerates at station2 and stops where it waits for ten seconds, then it accelerates again, until it reaches station3 where it decelerates and stops. The SIMULINK model is shown in figure (22).

The fuzzy memberships are shown in figure (23).

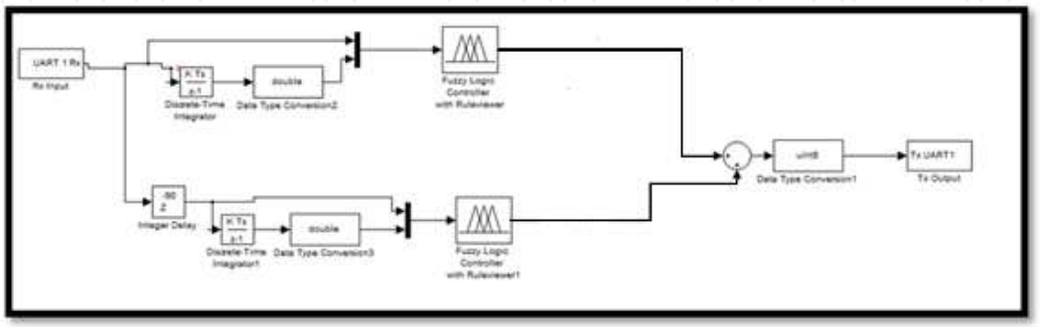

Figure 22. Three station controler

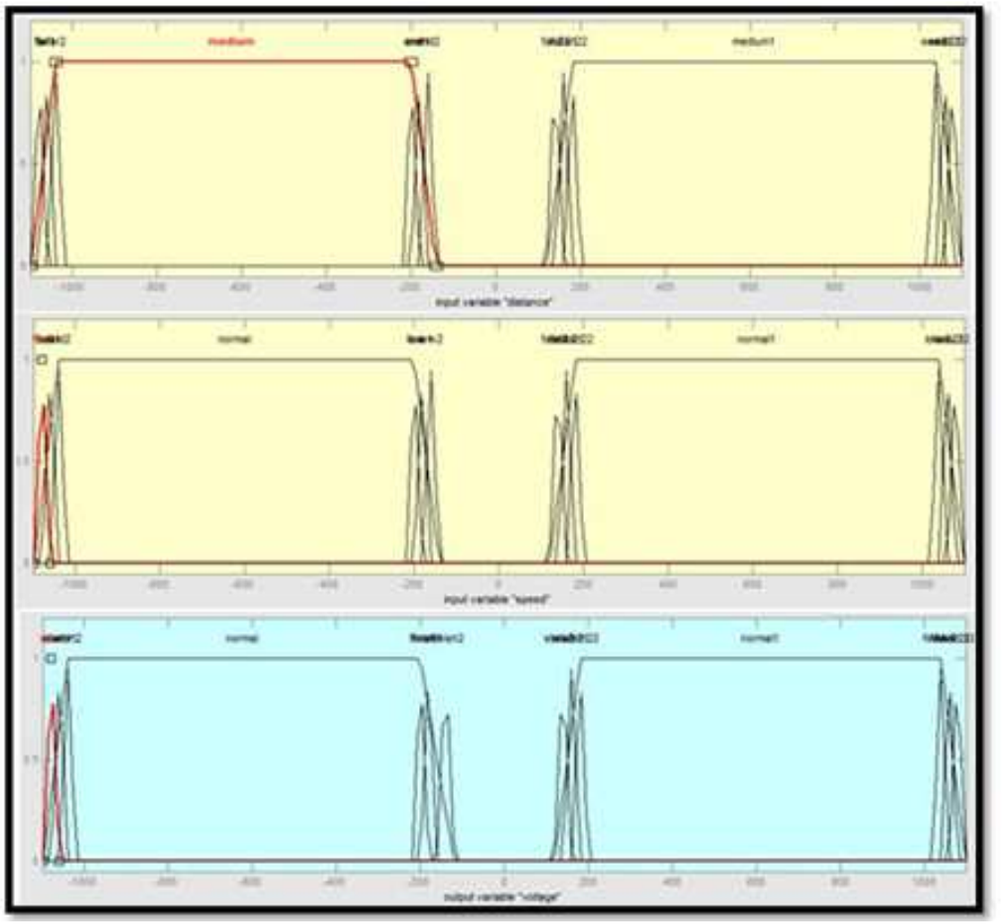

Figure 23. Input/output memberships for three stations

It could be noticed that each membership is divided into seven parts; acceleration at station1, then cruising between station1 and 2, then decelerating at station2, then waiting, then accelerating at station2, then cruising between station2 and 3 and finally stopping at station3. The trajectory rules are shown in figure (24).

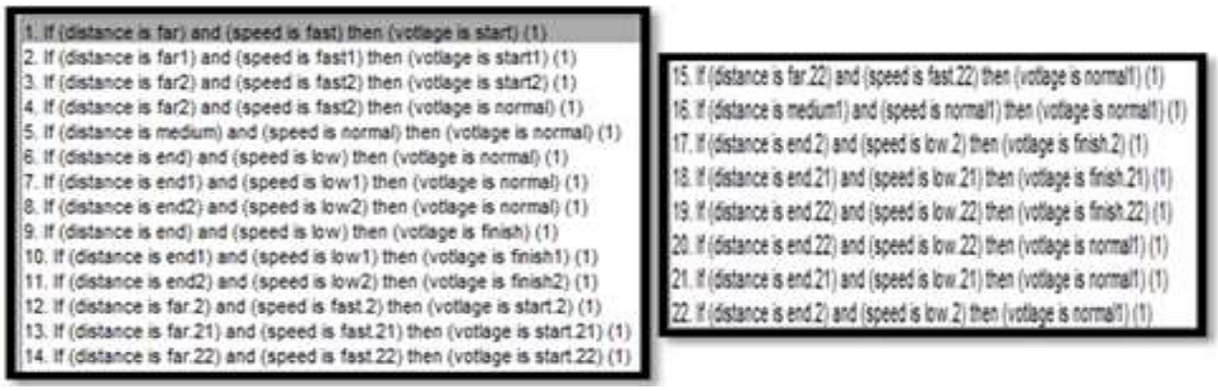

Figure 24. Fuzzy rules for three station system

The response of the system is shown in figure (25). To emulate a real system, the distances between station s to be different. Yet the response looks to be very smooth, if one ignores the steps due to discretization.

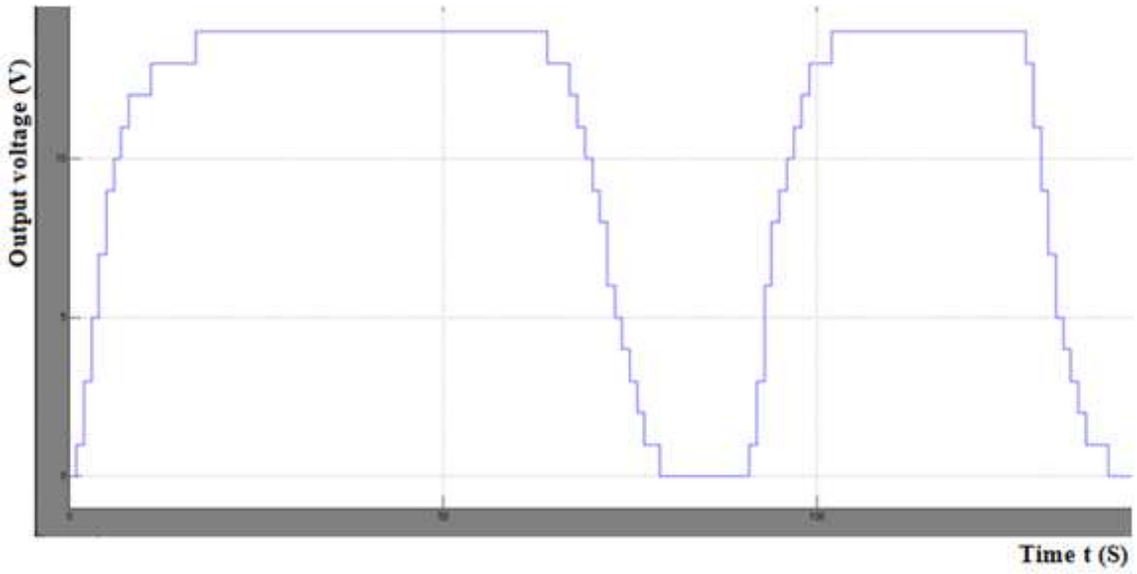

Figure 25. Three train station

The digital output from SIMULINK cannot drive the train, unless it is converted to an analogue signal. To do so, a 12 bit serial DAC is available from microchip which is the MCP4922. This converter has got a synchronous serial interface (SPI) with three connections; one is the serial data, one is the synchronizing clock and the third one is the chip select. On the output side, it

http://dx.doi.org/10.5772/57365

has two analogue channels A and B. In the application in hand, only channel A is used. The output of the converter is between 0 and 5V, so one requires to amplify this from zero to 12V. It was also noticed that the train starts moving only if the voltage reaches 4V. To do this we use the instrumentation amplifier INA114A together with a -5V regulator (LM7905T). The -5C is used to feed the negative rail on one side, and to obtain the exact required voltage of -4V at the inverting rail of the amplifier by using a potential divider. The output of the amplifier feeds an emitter follower to obtain enough current to drive the train electric motor. The voltage is applied to the train through the rails, so no extra arrangement is required. Figure (26) shows the output interfacing circuitry between the microcontroller and the train.

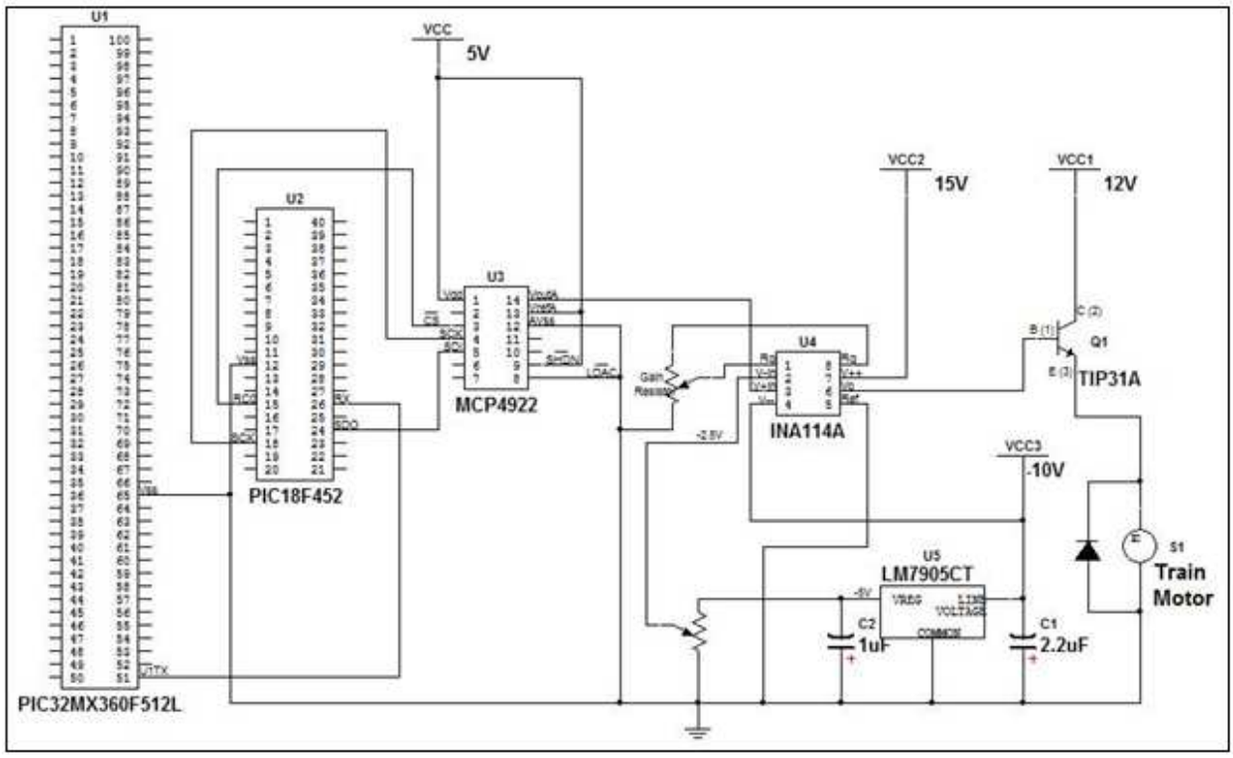

Figure 26. Output interface between the controler and the train

6. Conclusions

Though at the beginning MATLAB and SIMULINK were simulation tools to aid students and researchers to develop and simulate models. But there was a major drawback. That is it was difficult to interface those models to a real process. But in last few years things have changed where several companies have developed interfacing cards, together with their block sets (4& 5). Better still, in the last few years, block sets have been developed to program advanced microcontrollers such as dsPIC30 and dsPIC33 (6& 7) together with the powerful 32 bit PIC32 (7). This has renders the task of developing advanced control such as fuzzy logic, relatively

simple and interesting as well. Instead of going into the process of developing fuzzy control algorithm from scratch with all the problems which come with it in terms of time and bugs, it would be better to use on line a well established, easy to use algorithm such as SIMULINK based block set. This had made the simulating modeling straight forward (figures (5 & 6) and table (1)). The model representation was just click and drag type. (figures(18, 20 & 22)), so are the memberships (figures(14, 15 & 23)). One important factor, unlike other algorithms, the SIMULINK one could accommodate a big number of rules. (figures(16 & 24)).. In fact it could accommodate more than 100 rules.

On the design side, a remotely controlled train system was designed and tested. Though, the prototype is too small to be loaded to see its effect on the algorithm, nevertheless it has shown to be very effective. The beauty of the algorithm that we only bother at the transition periods (acceleration and decelerations) where special care has to be taken to fine tune the memberships and the rules no matter how many stations are used. Once the train enter the cruise trajectory, only a single trapezoidal membership is required This is demonstrated in simulation mode (figure (19)) though this looks to be very smooth, That is due to using continuous simulation. For real test, discrete integration was used. That is shown on the on line response (figures (21 & 25)) either for two stations or three stations respectively. This did not affect the response of the motor. Last but not the least, the compatibility of the electronics (figures (3 & 26)) such as the type of the microcontroller and its large stack and memory to accommodate all the if statements required by the fuzzy algorithm, plus the different ports such as the UART and SPI, to the serial converter and the signal conditioning circuitry to drive the train.

Though this chapter has shown the greatness of SIMULINK in rendering programming a very powerful and wonderful microcontroller such as the PIC32 relatively easy, it still faces a main challenge. That is how to monitor the speed online. Luckily, a call to the PIC serial port (USART) came to the rescue, where the port is interfaced to the computer serial port RS232, through a MAXIM max232 transceiver (7). Then a MATLAB input m-input function block is called. This block displays the speed response online as it shown in figures (21,25)

7. Alternative tools to SIMULINK

Practically there are alternative solutions other than SIMULINK such as LABVIEW and Fuzzy TECH. A company known as 3D micro Tools (3DμT) has developed a tool (10), compatible with NI LABVIEW, which generates PIC C code in similar manner as SIMULINK which could be fired into the PIC. The generated code could be fired into the PIC using the microchip MPLAB platform. However it is quite difficult at this stage to make a fair comparison between SMULINK based and NI LABVIEW based platforms, for the simple reason I have not tried yet. However, there is a will to do that in the near future. For the Fuzzy TECH, it looks though it is only implemented on certain old PIC microcontrollers such as PIC16C5x, PIC16Cxx and PIC17Cxx (11). As mentioned earlier, it looks though it is very difficult to implement a relatively accurate fuzzy algorithm on such controllers for the simple reason, that the stack memory is very limited. This in turn, makes the number of rules hence number of calls in fuzzy rules very limited as well. This may not produce accurate results.

http://dx.doi.org/10.5772/57365

Author details

Mostefa Ghassoul*

Address all correspondence to: mghassoul@uob.edu.bh

Chemical Engineering Department, College of Engineering, University of Bahrain, Bahrain

References

[1] http://www.kerhuel.eu/wiki/Download (This is the site where a demo SIMULINK block sets could be downloaded from).

[2] http://www.microchip.com/search/searchapp/searchhome.aspx?id=2&q=matlab (PIC32 is not included)

[3] Ghassoul, M. "pH control using MATLAB" Chapter in book MATLAB: A fundamental tool for scientist and engineering applications Volume 1 Edited by V.N.Katsikis, Intech PP243 – 268

[4] Ghassoul " Design of a Fuzzy System to Control a Small Electric Train Using Microchip PIC32 With the Aid of MATLAB Blocksets" IEEE Computer Science and Automation Engineering (CSAE) 2011 International Conference, Shanghai, Vol 03, PP 242-246

[5] Noor,S.B.M, Khor, W.C and Ya'acob,M.E. " Fuzzy logic control of a nonlinear pH neutralisation in waste water treatment" International Journal of Engineering and Technology,VOL. 1, No 2, 2004, pp 197-205

[6] Sang-Hoon Lee, Yan_Fan Li and Vikram Kapila " Development of a MATLAB-based graphical user interface environment for PIC microcontroller projects" Proceedings of the 2004 American Society for Engineering Education Annual Conference and Exposition Session 2220

[7] HUMUSOFT MF 624 Multifunction I/O card

[8] National Instruments PCI 6221 data acquisition card

[9] 3Dmicro Tools "Design, develop and download code on PIC microcontroller with NI LabView" This product was developed in collaboration with Prof. David Scaradozzi and supported by LabMACS, Laboratory of Modeling, Analysis and Control of Dynamical Systems, Department of Information Engineering, Universita' Politecnica delle Marche, Ancona (IT). January 2013

[10] Fuzzy TECH is a trademark of Inform GmbH and Inform Software Corporation

IntechOpen