FUZZY CONTROL APPLICATIONS DEVELOPMENT SYSTEM

Antonio Manzanedo[a], Jacobo Alvarez[b]

[a]Department of Systems and Automation Engineering, University of Vigo, Apartado oficial, 36200-VIGO, Spain.
[b]Department of Electronic Engineering, University of Vigo, Apartado Oficial, 36200-VIGO, Spain.

One of the major challenges of an educator is to introduce new technologies in such a practical way that undergraduates can compare them with existing technologies. To achieve this, most of the time it is necessary to develop an special equipment because there is no such commercially available system, or it is expensive or inadequate.

The goal of this work is to design and build an equipment that permits the development and implementation of fuzzy control applications to teach this new technique and compare it with classical control.

1. HARDWARE

The first decision was to use a general purpose microprocessor instead of a microcontroller to choose independently the peripherals such as A/D and D/A converters to achieve the required characteristics. The decision is very difficult since there are a lot of candidates, but in order to reduce the cost of the equipment, the Intel 8088 was chosen. This microprocessor has enough capabilities to be the core of our system, but the key reason to choose it was the possibility to recycle all the components of old personal computers (PCs) that are still stored in many warehouses of our University.

The system RAM memory is composed of two banks and can be configured between 64 Kbytes and 512 Kbytes depending on the RAM chips availability and our needs. We choose to use the same dynamic RAM chips that came with personal computers, so the 8237A, DMA controller, was included to refresh the data in memory.

The system EPROM memory is composed of two sockets for 2764 (8Kx8 each), one intended to store the boot and fuzzy monitor routines and the other prepared to store the application code in stand-alone operation.

Communications are implemented via RS-232 and RS-485 standards.

The interface with the analog world was achieved through an A/D converter MAX180 and two D/A converters MAX526, both from Maxims. The first one provides 8 multiplexed input channels with 12 bit resolution and 10 us. conversion time. The D/A converters have four 12 bit converters each and 3 us. settling time. They are fast enough for the intended applications in Mechatronics.

There are, of course, some other complementary circuits like the 8253 programmable timer, buffers, registers and 'glue' logic.

2. SOFTWARE

The boot and monitor routines that are located in the first EPROM of the equipment were developed in assembler language so they are optimized for both speed and code size. The software that runs on the personal computer, intended to communicate with the equipment, was written in Pascal and, at this time, the interface is DOS like, not very user friendly.

The definition of the inputs, membership functions, rules, etc. is done through an ASCII file with a format defined below.

3. OPERATION MODES

One of the objectives of this project was to design an equipment with the maximum flexibility so it could be used, not only to develop fuzzy control applications (the main goal), but to be used as a general data acquisition system or to implement non-fuzzy controllers. With this in mind, several working modes were developed, some computer attached and some stand-alone.

- **Mode 0**: local test. In stand-alone operation, the

equipment reads all analog and digital input channels and writes the values to the corresponding analog and digital output channels. This mode is intended for channel test only.
- **Mode 1**: memory read. In this mode, communication between a PC and the equipment is established and we can read any code or data stored in the RAM or EPROM modules. It is intended to test the communications link and the memory blocks.
- **Mode 2**: slave I/O operation. In this mode, we can use the equipment as an data acquisition system, having access to all inputs and outputs from another equipment. Working in this mode, we can implement in a computer any type of control algorithms over the external inputs and transmit the new outputs to the equipment.
- **Mode 3**: program load. In this mode, it is possible to load a new monitor program in the equipment RAM memory, so we can use this new program instead of the monitor program stored in the equipment EPROMs.
- **Mode 4**: stand-alone operation. In this mode, the equipment transfers the control to the monitor program stored in the equipment EPROMs. In that way, we can implement any kind of stand-alone controller.
- **Mode 5**: slave fuzzy operation. This is one of the most important modes, that allows the equipment configuration as a fuzzy control system via the RS-232 port. The computer sends the information about the membership functions of each input and output and the rules.
- **Mode 6**: not defined. For user purposes.
- **Mode 7**: stand-alone fuzzy operation. Instead of receiving the configuration through the RS-232 port, like in mode 5, this data is stored in the equipment EPROMs, so the system can work alone.

4. FEATURES

- The equipment can implement two or more independent controllers simultaneously, restricted to general characteristics.
- The analog inputs are fuse protected.
- The maximum number of inputs are 8, either analog or digital.
- The resolution of the analog inputs is 10 bits.
- Each input can have up to 8 fuzzy sets or regions.
- Each region must be defined by a membership function of any polygonal shape.
- The degree of membership can take values between 0 and 127.
- Each rule can have up to 256 (the number of inputs) complex antecedents. A complex antecedent is one formed by two or more simple ORed antecedents. A simple antecedent is formed by up to 8 ANDed antecedents where each antecedent (pair input-region) has to be related with any different input variable. A multiple rule is one with complex antecedents. An individual rule is one with simple antecedents.
- Each application permits a maximum of 256 individual rules or a combination of individual and multiple rules.
- The defuzzyfication method is the centroid one.

5. SOLUTIONS

To obtain the value of each output of a fuzzy controller in a given instant, implies a heavy computation load. This work has to be done in a short time to achieve the performance of a real-time controller, even in slow systems, such as motors, etc..

Another possibility is to tabulate all the values that can be obtained from this calculation. This is possible because the inputs can only have a finite number of values, depending on the number of resolution bits of the A/D converter, and there are a finite number of membership function, previously defined for each application, and a finite number of rules. However, it requires a high amount of memory since the table size depends exponentially on the number of inputs, the number of resolution bits and lineally with the number of outputs. For each output it is necessary a table which size will be $(2^{\text{nº of bits}})^{\text{nº of inputs}}$.

To reach a compromise between computation power and memory size needed, we decided to use a semi-tabular method, that is composed by the following steps (figure 1).

1) With the inputs, their regions and each region membership function defined, we build a **membership table** that, for every possible value of the input, gives the degree of membership to each region (1 byte). So, the fuzzyfication method consists only in accessing this table. The table size is given by the maximum number of inputs (8), the maximum number of regions per input (8) and the possible values of each input (1024 = 10 resolution bits). So, the size is 8x8x1024 = 64 Kbytes.

2) With the outputs, their regions and each region membership function defined, we build an **area table** that, for every possible value of the output, gives the

total area below that value in the membership function and the momentum related to the origin. The table size is given by the maximum number of outputs (8), the maximum number of regions per output (8) and the possible values of the degree of membership (128) and the bytes needed to store the area and momentum (8). So, the size is 8x8x128x8 = 64 Kbytes.

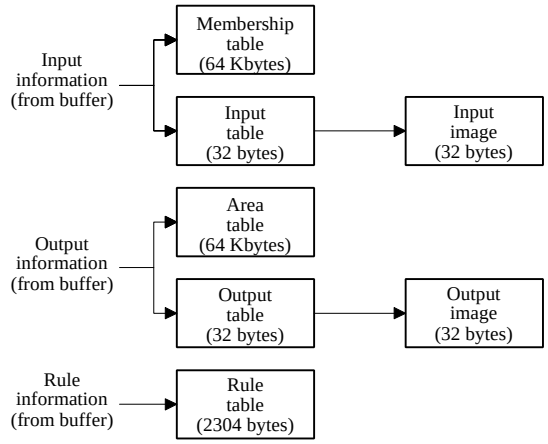

Figure 1. Table calculation.

3) With the rules, we build a **rule table**. The table size is given by the maximum number of individual rules (256) and the maximum number of antecedents (8) that form a simple antecedent, the number of bytes needed to represent the region involved in each antecedent (1) and the region of the consequent of the rule (1). So, the total size is 256x(8x1+1x1) = 2304 bytes.

4) With the definition of the inputs, we build an **input table** that gives information about the input type (analog, digital, unipolar, bipolar), the number of regions described, etc.. This table is 32 bytes wide. When the application is running we build another table called **input image**, that contains the input type and the present value of each input and it is 32 bytes wide, too. In the same way, we build the **output table**, auxiliary for calculating the value of the outputs while the system is running, and the **output image**, each 32 bytes wide.

All these tables, except the input and output images, do not change their contents once defined for a specific application. When we initiate the process, that is, the fuzzy control of the target system, first the table values are calculated and memorized in the equipment RAM and then, each processing cycle is composed by the following steps (figure 2).

1) The values for the inputs are obtained, storing them in the input image.

2) The membership table is used to obtain the degree of membership to each region for every input.

3) With these degrees of membership (x) and their complementaries (127-x) we build a **membership vector**, 128 bytes wide.

4) With this vector, the rules are applied through the rule table, obtaining the consequent (pair output-region) and calculating the value of the degree of membership for this rule. This information is stored in the **level table**.

5) The defuzzyfication is achieved by adding the areas and momentums related to the fired rules. The values for each output are calculated using the information of the level and area tables and are stored in the **auxiliary output table**, 128 bytes wide.

6) The final value of each output is obtained dividing total momentum by total area and storing the results in the output image. These values are written to the outputs in the end of the cycle.

Although integer arithmetics is used to simplify the calculation, the maximum errors do not reach 3 units in 1024, this is, 13 mV. in a range of 0-5 V.

6. PRACTICAL RESULTS

Using the maximum number of rules, 64, 4 inputs with an average number of regions per input of 4, the total cycle time is below 14 ms., this is, about 70 cycles per second, that is enough for many typical mechatronics applications.

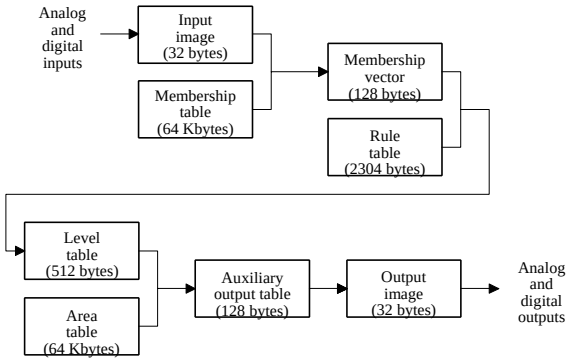

Figure 2. Cycle execution

7. TUTORIAL EXAMPLE

We have chosen a modern home application, water temperature and flow control to show the powerness and simplicity of this equipment. As you can see in figure 3, there are two valves to control the flow and and two sensors to measure the temperature of cold and hot water and one more sensor to measure the flow of the resulting mixture.

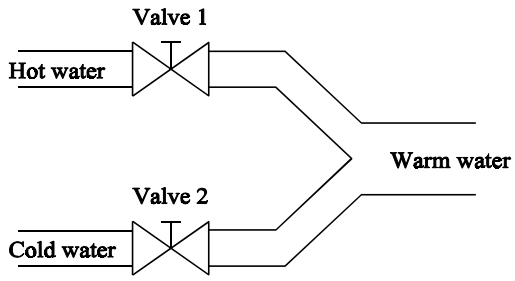

Figure 3. Home plumbing.

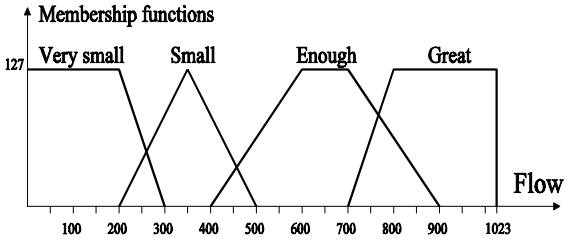

Figure 4. Membership functions for Water Flow.

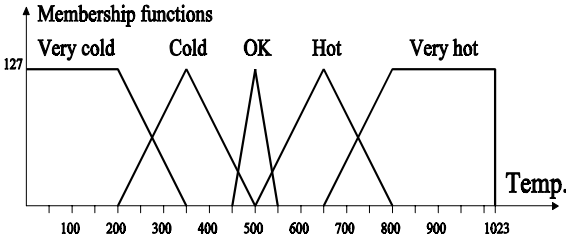

Figure 5. Membership functions for Water Temperature.

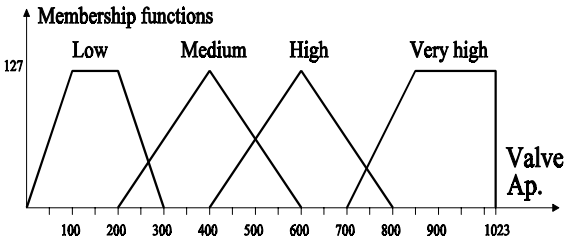

Figure 6. Membership functions for Valve Aperture.

The memberships functions that we have defined for the flow (F), the temperature (T) and the valve aperture (V) are in figures 4, 5 and 6. The rules in table 1 represent the valve aperture (output) depending on the values of flow and temperature (inputs). The value "nothing" means that the last value is not changed. With these rules, the processor

Table 1
Valves aperture (fired rules)

	Flow			
	Very small	Small	Enough	Great
Temperature				
Very cold	Cold nothing Hot very high	Cold nothing Hot very high	Cold medium Hot very high	Cold low Hot high
Cold	Cold nothing Hot high	Cold nothing Hot high	Cold medium Hot high	Cold medium Hot nothing
OK	Cold very high Hot very high	Cold high Hot high	Cold nothing Hot nothing	Cold medium Hot medium
Hot	Cold high Hot nothing	Cold high Hot nothing	Cold high Hot medium	Cold nothing Hot medium
Very hot	Cold very high Hot nothing	Cold very high Hot nothing	Cold very high Hot medium	Cold high Hot low

executes 150 cycles/s, so the response time of the controller is optimal and negligible to the potential user.

We have tried some other applications like direct current motor velocity control and simulation of the Sugeno's mobile and the results are very satisfactory in precission and speed.

8. CONCLUSIONS

To summarize this work, we can stand out the following characteristics:
- Compact, strong and easy to use fuzzy application development equipment.
- Reduced hardware costs since many components are recovered from old PCs.
- The computational method uses tables as well as integer calculation, achieving a reasonably speed with a low amount of memory.
- Flexibility is one of the keys, so this equipment can be used as a data acquisition system or as any other (non-fuzzy) control application development system.
- It works on both slave (computer attached) and stand-alone modes.
- The code is written in assembler so it is optimized.

In figures 7, 8 and 9 you can see the fuzzy application development equipment (VER FOTOS PÁGINA WEB).

9. FUTURE DEVELOPMENTS

- Development of Windows-based interface.
- Reduce the board size, using programmable circuits such as FPGAs.
- Increase the computation speed, substituting the 8088 for another faster microprocessor.

10. BIBLIOGRAPHY

[BRUBAKER 92] BRUBAKER, David I., SHEERER, Cedric, "Fuzzy-logic system solves control problem", *EDN*, June 18, 1992.

[CONNER 93] CONNER, Doug, "Designing a fuzzy-logic

control system", *EDN*, March 31, 1993.

[CUBILOT 95] Cubilot Rodríguez, Marta, "Aplicación de la lógica borrosa al control de velocidad de un motor de corriente continua", Proyecto Fin de Carrera, Universidad de Vigo, Vigo, 1995.

[LEGG 93] LEGG, Gary, "Microcontrollers embrace fuzzy logic", *EDN*, September 16, 1993.

[MANZANEDO 97] Manzanedo García, Antonio, "Equipo para el desarrollo de aplicaciones de lógica borrosa", Proyecto Fin de Carrera, Universidad de Vigo, Vigo, 1997.

[NEURALOGIX 91] Fuzzy Microcontroller Development System Manual, American Neuralogix Inc., 1991.

[SUGENO 85] Sugeno, M., Nishida, M., Fuzzy control of model car, *Fuzzy Sets and Syste ms 16*, pp. 103-113, 1985.