Preprints of the 2012 20th Mediterranean Conference on Control & Automation (MED), Barcelona, Spain, July 3-6, 2012

Microcontroller Based Fuzzy-PI Approach Employing Control Surface Discretization

Jasmin Velagić, Muhamed Kurić, Edin Dragolj, Zlatan Ajanović & Nedim Osmić
Department of Automatic Control & Electronics
Faculty of Electrical Engineering at the University of Sarajevo
Zmaja od Bosne, 71000 Sarajevo, Bosnia and Herzegovina
Email: jvelagic@etf.unsa.ba, muhamed.kuric@etf.unsa.ba, edindragolj@hotmail.com, ajanovic.zlatan@gmail.com, nosmic@etf.unsa.ba

***Abstract*—This paper proposes an approach of digital controller design on microcontrollers based on control surface disretization. As experimental setup a simple system composed of a DC motor, the PWM drive and a encoder was built. Identification of this simple system was done using the first order linear model and the Hammerstein-Wiener model. The quality of these models was compared based on the fitness function and their ability to cope with system nonlinearities. A discrete PI speed controller was designed based on the Hammerstein-Wiener model and tuned by the Gauss-Newton optimization algorithm. The Fuzzy logic controller was designed on the basis of the PI controller behaviour with genetic algorithm parameter tuning. The control surface of the Fuzzy logic controller was discretized for microcontroller implementation. Both controllers were implemented on the Arduino Mega platform and their control performances were tested and compared.**

Keywords — DC Motor Control, Hammerstein-Wiener Identification, PI Controller, Fuzzy-PI controller, Genetic Algorithm, Arduino Mega, Time-Space Tradeoff, Control Surface Discretization.

I. Introduction

Microcontrollers have become quite affordable in recent years, which is mostly due to the rapid development of integrated electronics. This is a primary reason why mechatronic systems often contain microcontrollers as control units. Each control unit is used for multiple concurrent tasks, including control algorithms, field interface, data storing, communication with peripheral electronic units, etc. Tasks like control algorithms are periodic tasks with a deadline, thus requiring a real time implementation [1].

The software implementation of concurrent tasks which execute in real time is a challenging programming problem considering typical computational and storage constraints on most 8 & 16 bit microcontrollers. Other problems involving software design are changes that reflect on the minimal execution time of tasks (implemented via timer interrupts) which can break software modularity. Such tasks are mostly control algorithms, communications and visualisations with control algorithms using up most execution time [2].

There are two typical approaches in microcontroller based control algorithm implementation with 8 bit microcontrollers. The first approach being a direct implementation of the control algorithm using numerical representations of the controller mathematical model. This approach is mostly used for simple control algorithms, such as the classical PID control algorithm [3] and its improvements [4, 5]. The only advantage of this approach being simple software implementation of the control algorithm. The major drawbacks of this approach being numerical calculations which are prone to errors on low performance hardware [6] and produce negative controller performance [7]. Every change in control algorithm can result in increased execution time beyond tolerated limits which makes it difficult to maintain software designs based on this approach.

The second approach consists of an implementation of the control algorithm based on a partially utilized lookup tables and soft control algorithms (mostly Fuzzy logic and neural network controllers). Partially utilization of the lookup table is a consequence of the fact numerical calculations of the control output are inevitably. The control output is computed from a storage table which memorizes some aspects of the controller, e. g., membership functions [8]. The possibility to mimic other control algorithms [9, 10] allows a unified software implementation strategy for any type of controller. This implementation can be more advanced than the direct approach, thus resulting in better control performance, while maintaining the same numerical and software maintenance shortcomings.

Instead of using a lookup table for solely memorizing some aspects of the controller we propose in this paper an indirect approach of the Fuzzy-PI controller design based on a lookup table. In order to use this approach it is necessary to obtain a good simulation model of the controlled object. The Controller design and parameter tuning procedure is based on the obtained simulation model or online using real time hardware in on-line mode. Design and tuning results can be validated through on-line experiments on the physical controlled object. The designed controller behaviour is mapped into a suitable form which is adequate to create a lookup table based on control surface discretization. This mapping can be accomplished by using typical optimization algorithms, i. e. gradient methods or genetic algorithms to obtain a Fuzzy-PI

978-1-4673-2529-5/12/$31.00 ©2012 IEEE

controller which mimics the PI controller [11].

The discretized control surface point values of the Fuzzy controller are stored into the microntroller memory and the its output is generated based on indexing or searching the lookup table. This approach ensures an optimal execution time, which implies minimal controller sample time due to the fact that the lookup table is presorted and precomputed and thus achieving a classical time-space tradeoff [12]. Based on typical resolutions of analog inputs and outputs of common 8 bit microcontrollers the lookup table can have a fixed size. This provides a robust way to change the control algorithm without compromising the modularity of the microcontroller software design (control algorithm changes are not reflected in execution time) and achieving reuseability of the implemented software. In order to describe the proposed approach this paper analyses a case study considering the PI speed controller implementation of a simple motor system.

The paper is organized as follows. In section 2 the problem statement with proposed control system description is given. The plant identification using linear and Hammerstein-Wiener models is given in section 3. Section 4 describes controller design for the Arduino Mega microcontroller, while section 5 gives analysis of simulation and experimental results. Conclusions and further research guidelines are discussed in the final section.

II. Proposed Control System Description

A. Problem Statement

In the traditional control approaches based on the conventional PI controller, the controller output u is a function of the current error value e and time instant t and can be expressed as

$$u = u(e(t), t). \tag{1}$$

Numerical representations of (1) include negative effects, like saturation that causes negative dynamical behaviour and oscillations due to calculations of derivatives.

In order to apply the proposed approach it is necessary to transform (1) into its estimate

$$\widehat{u} = \widehat{u}(\mathbf{r}(t)), \tag{2}$$

where $\mathbf{r}(t)$ is the input vector of the controller (the controller output itself is time-invariant function in this model). The geometrical interpretation of (2) represents a surface or generally a hypersurface. The surface discretization based on the microcontroller input and output resolution is required in order to store this surface efficiently in the microcontroller.

The control law represented by (2) can be efficiently modelled using soft computing methodologies, like neural networks and fuzzy logic controllers. Fuzzy logic controllers are inherently modelled as control surfaces and represent the optimal solution for (2). This is the primary reason why a fuzzy logic controller model was chosen in this paper.

Transforming $\widehat{u}$ into u can be considered as an optimization problem described by

$$\min J(e_{u,\widehat{u}}), \tag{3}$$

where J is a performance index (criterial function) of the controller output sequence error $e_{u,\widehat{u}}$. The controller output sequence error is given by

$$e_{u,\widehat{u}} = \Psi(u(e(t), t)) - \Psi(\widehat{u}(\mathbf{x}(t); \boldsymbol{\Xi})), \tag{4}$$

where Ψ is the plant function and $\boldsymbol{\Xi}$ is the tuning vector of the function $\widehat{u}$. The tuning vector parameterizes the controller structure. Optimization of this vector based on (3) and (4) produces a better representation of the PI controller by the Fuzzy-PI controller. The exact structure of $\boldsymbol{\Xi}$ is explained in section 4, in more details.

Minimizing J gives a good matching between the control functions u and $\widehat{u}$. The optimization problem described by (3) can be solved using various approaches, i. e. conventional nonlinear programming, genetic algorithms, etc. The simple genetic algorithm is chosen for optimal estimation of $\boldsymbol{\Xi}$ due to its efficiency for this problem class [13].

B. System Block Scheme

The system block scheme is shown in Fig. 1. It contains a simple system (plant) consisting of the DC motor, the PWM chopper power drive, the incremental encoder with frequency to voltage converter (KA 331 integrated circuit), the Arduino Mega 1280 microcontroller system and PC. The Arduino Mega microcontroller serves as the plant controller. Two different control algorithms, the conventional PI and th Fuzzy-PI controller, were implemented in this paper. The Fuzzy-PI controller is designed based on the lookup table obtained using surface discretization. Communication between PC and Arduino Mega Microcontroller was established using the USB protocol and served for data acquisition and setpoint changes. Based on the problem statement in this section the control block scheme shown in Fig. 2 was proposed to map the fuzzy logic controller into a PI controller. The input sequence is propagated through both controllers and the plant outputs Ψ_u and $\Psi_{\widehat{u}}$, which appeared in (4), are obtained. The output sequence error $e_{u,\widehat{u}}$ is calculated and the performance index J is evaluated based on its value. The genetic algorithm, based on the value of the performanc index J, tunes the parameter vector $\boldsymbol{\Xi}$ of the fuzzy logic controller. This process is then repeated for a fixed number of epochs. The control surface discretization of the obtained Fuzzy-PI controller is used for the microcontroller lookup table generation. This is explained in section 4, in details.

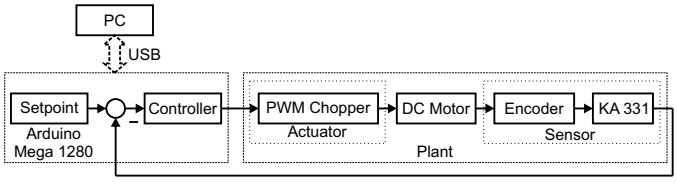

Fig. 1. System block scheme

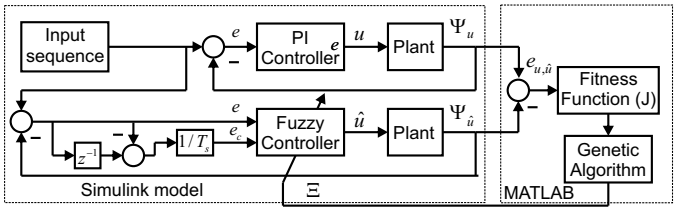

Fig. 2. Control block scheme of the cloning process

III. System Identification

A. Experimental Setup

The experimental setup for verifying the proposed approach is shown in Fig. 3. It contains: the two quadrant PWM chopper, the DC motor and an incremental encoder with frequency to voltage converter. The DC motor has 3600 rpm nominal speed, 24 V nominal voltage and 2.2 A nominal current. The PWM chopper requires less components and is easier controlled with a microcontroller than a typical linear voltage amplifier. The two quadrant chopper is capable to control the motor in one direction while having better braking utilization than an one quadrant chopper. Power FET's have typically 20 V maximum gate voltage and based on that constraint the output voltage of 20 V was chosen (implying that the maximal motor speed with this experimental setup is 3000 rpm). The maximal PWM frequency of 150 Hz was chosen based experimentally on the worst case scenario (steady state driver temperature of 46 °C was measured in that case). The incremental encoder with frequency to voltage converter acts as sensor converting input motor speed range of 0 - 3000rpm lineary into 0 - 4.096V.

B. Identification Approaches

Idenfication approaches can be divided into three categories white-box, grey-box and black-box. The white-box approach consists of physical modelling of the system and creates an accurate system model. The problem with this approach is that it is only applicable on simple systems. The grey-box approach creates an apriori model with parameters which are estimated in the identification process. To use this method it is required to determine an appropriate model which resembles the system dynamic. The black-box approach is particullary used when apriori information of the model is unknown. Model interaction is possible and due to that more detailed model classifications exist [14].

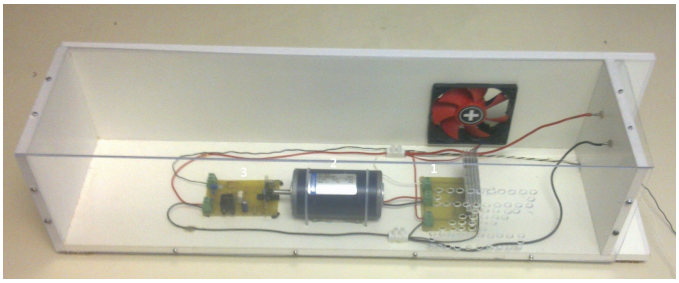

Fig. 3. Experimental setup

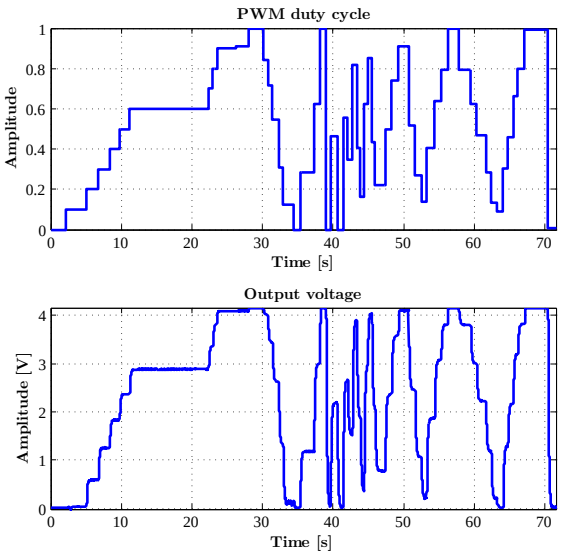

Fig. 4. The Measured data set used for identification

Identification of models with several components can be either coupled or decoupled. The coupled indentification is performed on the system in whole, while the decoupled identification is used to identify each component of the system seperatly.

We chose two grey-box approaches to identify the all system components with one model using the System Identification Toolbox, which is a part of the MATLAB software package [15]. Both approaches use the same set of measured data shown in Fig. 4 sampled with T_s = 0.1 ms. The input sequence was randomly generated consisting of step sequences that cover most duty cycles of the actuator. The dynamic of the sequence was experimentally chosen so that the motor can follow the input dynamically.

C. Linear Model Identification

A DC motor transfer function with armature voltage as input and speed as output can be expressed by the 2nd order transfer function [16]. By ignoring the voltage-current dynamic it is possible to obtain the 1st order transfer function. This approximation is used as grey-box choice and the resulting identified transfer function is given by

$$G(s) = \frac{K}{T_f s + 1}, \tag{5}$$

where $K = 4.3657$ and $T_f = 0.15127$.

D. Hammerstein-Wiener Model Idenfitication

However, the system exploits some degree of input and output nonlinearities. The input nonlinearity is caused by the motor dead zone, and output nonlinearity is caused by the saturation of the frequency to voltage converter. They are both statical nonlinearities, which represent time invariant nonlinearities. We used the Hammerstein-Wiener for the identification of the simple system in order to include

the mentioned nonlinearities into the system model. The Hammerstein-Wiener model is composed of the linear model, input and output nonlinearities. The mathematical representation of Hammerstein-Wiener model is given by [17]

$$y(t) = g(G(q)f(u(t))), \tag{6}$$

where f and g are estimated functions of input and output nonlinearity, respectively. The discrete transfer function of the linear part of the model being defined as $G(q)$.

The following regressor structure was chosen as $G(q)$

$$G(q) = \frac{b_1 q^{-1}}{1 - f_1 q^{-1}} u(t) + e(t), \tag{7}$$

where $u(t)$ is the system input, $e(t)$ is noise, with estimated parameters $b_1 = 0.003833$ and $f_1 = 0.9993$. The nonlinear part of (6) contain the motor dead zone and saturation. Estimated dead zone and saturation are 0.08935 duty cycle and 4.096V, respectively.

E. Model Comparison

The Measured and simulated system response for both models on the identification input excitation are depicted in Fig. 5. From those responses can be concluded that the Hammerstein-Wiener model resembles the measured response more than the linear model (this is reflected on model fitness shown in the legend). Both models have the same dynamic as the measured output with differences only in output values which are very close to the dead zone and saturation, with the Hammerstein-Wiener model having less differences (almost none) than the linear model. This is a consequence of better statical nonlinearity characteristic of the Hammerstein-Wiener model shown in Fig. 6. The difference in the Hammerstein-Wiener model characteristic was caused by the noise in the measured system response.

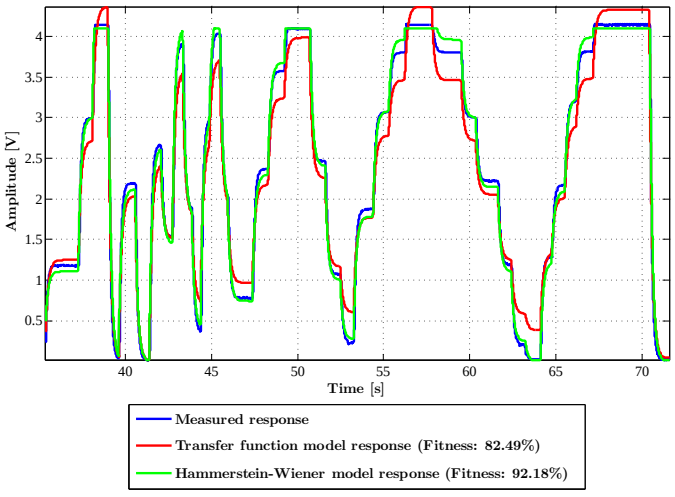

Fig. 5. Measured and simulated system responses

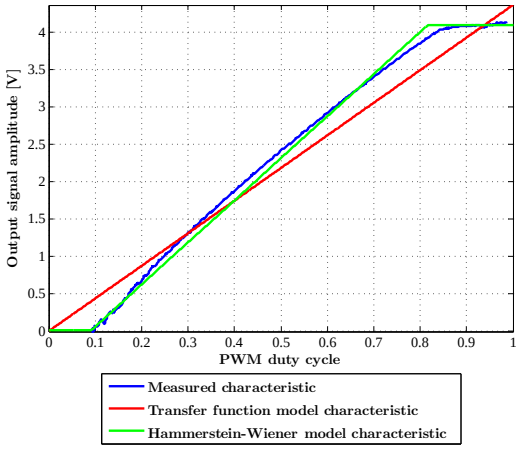

Fig. 6. Static characteristics of the obtained models

IV. Controller Design

A. PI Controller Design

The PI algorithm is oftenly used to control motor plants with a microcontroller. The implementation is based on the discrete PI controller which is obtained by discretizing the continuous PI algorithm. The discrete PI algorithm can be expressed as

$$u(kT_s) = K_p e(kT_s) + K_i T_s \sum_{i=0}^{k} e(kT_s); k, i \in \mathbb{N}_0, \tag{8}$$

where e is the controller input, u the controller output and T_s the sample time of the controller. K_p and K_i are controller coefficients, which need to be tuned. Typicial tuning methods can be categorised into formula-based, rule-based and optimization-based. Formula-based methods first identified the characteristics of the plant and then perform a mapping (like the Ziegler-Nichols formula). Rule-based methods are often used in adaptive control, but can be quite complex and ad hoc. Optimization-based methods are often applied offline or on very slow processes, using a conventional search method such as least mean squares [18].

The tuning of the parameters of the PI controller was done using the Signal Constraint block in Simulink. The Gauss-Newton algorithm was used as optimization method [19] with step signal as input reference. Estimated optimal parameters of the PI controller are

$$K_p = 10.741; K_i = 18.561. \tag{9}$$

Microcontroller implementation was done using (8) and (9) in combination with a T_s timer interrupt.

B. Fuzzy-PI Controller Design

The purpose of the fuzzy logic controller, as described before, is to mimic the designed PI controller. We propose the following fuzzy logic controller structure with adequate parametrization, described with the vector of tunable parameters (Ξ). The proposed Fuzzy PI controller structure is

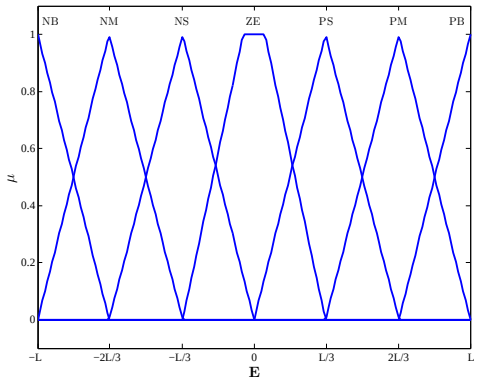

Fig. 7. Error membership function

similar to standard approach but does not contain an output integrator. The motivation of this decision is to avoid integrator caused software shortcomings. This way negative effects on the controller behaviour caused by overflow and wind up were avoided. A similiar Fuzzy logic controller design is described in [20]. Controller inputs are error e and error change e_c defined as

$$e(kT_s) = r(kT_s) - y(kT_s) \quad (10)$$

$$e_c(kT_s) = \frac{e(kT_s) - e(kT_s - T_s)}{T_s}; k \in \mathbb{N}_0, \quad (11)$$

where r is the current setpoint. Linguistic variables used in the fuzzy logic controller are E and E_c (with their membership functions shown in Fig. 7 and Fig. 8.) for crisp values e and e_c, respectively. The Parameters L and Q are used to parametrize crisp value ranges for error (e) and error change (e_c) signals, respectively. This selection of membership functions provides certain advantages in terms of uniform overlap of input values and reduced number of parameters which will be determined by genetic algorithm.

The set of Fuzzy logic controller rules is presented in Table I and is derived by examining characteristical points from the closed-loop system response, as described in [20]. 7 membership functions were chosen based on experimental evidence and to illustrate the efficiency of the proposed approach for fairly complicated controllers. Considering the control systems' simplicity 5 or even 3 membership functions are acceptable. The fuzzy logic controller is a Takagi-Sugeno type and so it implies that output membership functions are constants, which are suitable for parametrization. Parameters $\beta_i (i = 1..5)$ are used for parametrization, while β_1 and β_5 are fixed boundary values of the controller output, being 0 and 1 respectively. Determination of $\boldsymbol{\Xi}$ is based on the approach shown in Fig. 2 with fitness function J given as

$$J = ln(1 + \sum_{i=1}^{k} e^2_{u,\widehat{u}}), \quad (12)$$

where k is the plant output sequence size. This function was chosen because it provides fast convergences for huge and small errors alike.

Used settings for the genetic algorithm are: population size - 40, maximum generations count - 20, crossover fraction - 0.8, elite count - 2. Genetic algorithm yielded the following tunable vector value

$$\boldsymbol{\Xi} = \begin{bmatrix} \beta_2 & \beta_3 & \beta_4 & L & Q \end{bmatrix} \quad (13)$$
$$= \begin{bmatrix} 0.093 & 0.052 & 0.981 & 0.061 & 5131.671 \end{bmatrix}.$$

The Fuzzy logic control surface based on parameters given with (13) is shown in Fig. 9. The surface resembles a bang bang controller which is a consequence of the PI controller from which it was cloned. The output sequence of the PI controller was propagated through the plant and saturated output was obtained, which implies that the PI controller resembles a bang bang controller when driving an output saturated plant.

For faster microcontroller execution time the error change e_c was scaled into the range $[-1, 1]$, which implies that error e and scaled error change e^s_c have the same input range. This fuzzy variable input range matches the voltage range of $[-4.096, 4.096]$, thus to convert the control surface input

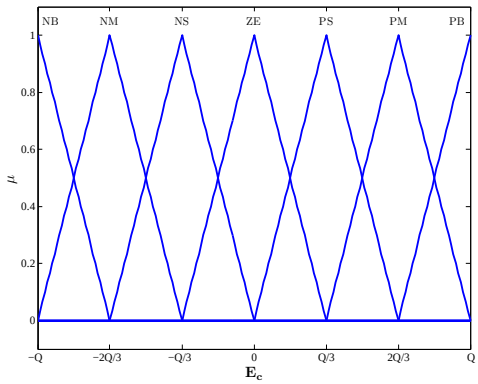

Fig. 8. Error change membership function

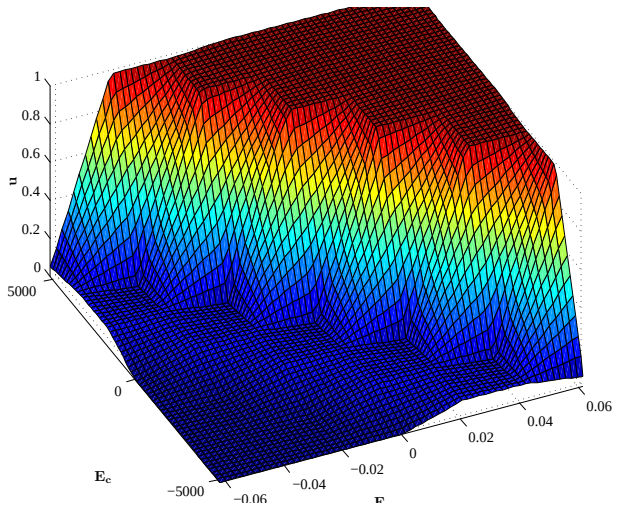

Fig. 9. Fuzzy logic controller control surface

TABLE I
FUZZY-PI CONTROLLER RULES

E/E_c	NB	NM	NS	ZE	PS	PM	PB
NB	β_1	β_1	β_1	β_1	β_2	β_2	β_3
NM	β_1	β_1	β_1	β_2	β_2	β_3	β_4
NS	β_1	β_1	β_2	β_2	β_3	β_4	β_4
ZE	β_1	β_2	β_2	β_3	β_4	β_4	β_5
PS	β_2	β_2	β_3	β_4	β_4	β_5	β_5
PM	β_2	β_3	β_4	β_4	β_5	β_5	β_5
PB	β_3	β_4	β_4	β_5	β_5	β_5	β_5

variables into signed digital values we used

$$\begin{bmatrix} e_{int} & e_{cint} \end{bmatrix} = \frac{4.096}{1024} \begin{bmatrix} e & e_c^s \end{bmatrix} \tag{14}$$

which is obtained by simulating 10 bit analog-digital conversion on a 5 V scale (which is the default Arduino Mega configuration). These quantized error and error change values are used as lookup table indexes for 8 bit quantitized output. The output quantization error (which is generally under one quant) is shown in Fig. 10. Based on this quantization a lookup table with 13×431 elements, which containts non-saturated Fuzzy-PI controller output, was obtained. On the used Microcontroller model approximately 23 lookup tables of this size can be stored. It is even possible to use the more expensive Arduino Mega 2560 model that has twice as much programming memory, so it is possible to store 46 of those lookup tables. This makes the Arduino Mega an ideal choice to implement complex control algorithms based on the proposed approach.

To calculate the Fuzzy-PI controller we propose the algorithm whose pseudo code is illustrated in Algorithm 1. In each iteration the digital error e_{int} and digital error change e_{cint} values are calculated, then their respective lookup table indexes e^i and e_c^i are calculated. Those indexes are needed to satisfy the boundary

$$(e^i, e_c^i) \in [-L_q, L_q] \times [-Q_q, Q_q], \tag{15}$$

where L_q and Q_q respectively represent the digital values of L and Q given by (14). Typical programming languages do not support negative indexes, which implies that index offsets are required. In this term the offset values only depend on the control surface of the concrete controller. The Controller output is produced by simply indexing a 2-dimensional array which represents the control surface S.

V. SIMULATION AND EXPERIMENTAL RESULTS

Fig. 11 shows simulated responses of the PI and Fuzzy-PI controller. We see that the Fuzzy-PI controller exhibits better dynamical performance in comparison to the conventional PI controller. This is caused by the fact that the fuzzy controller output is a time-invariant function. Also the saturation effect that is a consequence of numerical PI controller implementation slows down its dynamics. This effect can be seen when negative change of reference occurs in Fig. 12. The Fuzzy-PI controller has a maximal static error of 1.25% on higher

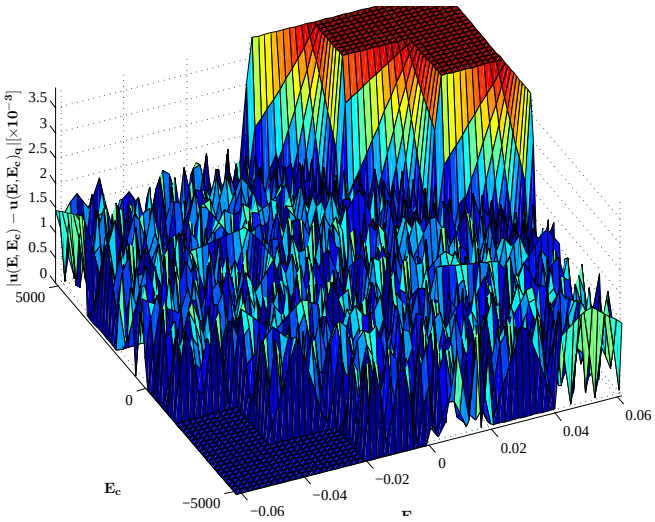

Fig. 10. Control surface quantization error

Algorithm 1 Proposed Fuzzy-PI controller algorithm

Require: That $e_{int}(k)$, $e_{int}(k-1)$ and $e_{cint}(k)$ are 16bit integers.

Calculate $e(kT_s)$ and $e_c(kT_s)$:
$e_{int}(k) \leftarrow V_{ref} - V_k$
$e_{cint}(k) \leftarrow e_{int}(k) - e_{int}(k-1)$

Calculate control surface lookup table indexes:
$e^i \leftarrow e_{int} + e_o^i$
$e_c^i \leftarrow e_{cint} + e_{co}^i$

Verify e^i index upper boundary:
if $e^i > e_u^i$ **then**
  $e^i \leftarrow e_u^i$
end if

Verify e^i index lower boundary:
if $e^i < e_{lb}^i$ **then**
  $e^i \leftarrow e_{lb}^i$
end if

Verify e_c^i index upper boundary:
if $e_c^i > e_{cub}^i$ **then**
  $e_c^i \leftarrow e_{cub}^i$
end if

Verify e_c^i index lower boundary:
if $e_c^i < e_{clb}^i$ **then**
  $e_c^i \leftarrow e_{clb}^i$
end if

Set controller output $u(kT_s)$ via control surface S:
$u(k) \leftarrow S(e^i, e_c^i)$

Store $e(kT_s)$ for the next iteration:
$e_{int}(k-1) \leftarrow e_{int}(k)$

speeds. This can be considered as a disadvantage. The PI controller does not have a statical error because the integral

TABLE II
ARDUINO CONTROL ALGORITHM EXECUTION TIMES

Controller	min [μs]	avg [μs]	max [μs]
Fuzzy-PI	0	3.37	4
PI	4	6.56	8

component reduces steady state errors.

Fig. 12 shows real-time responses of the PI and Fuzzy-PI controller obtained without external disturbances acting. These are identical beside that the real-time responses are covered by process noisse. The noise is produced by the PWM driver and could be eliminated using a median filter. The PI controller has a problem to cope with the motor dead zone. The statical error of the Fuzzy-PI controller became smaller due to saturation differences in the Hammerstein-Wiener model shown in Fig. 6.

In order to benchmark the execution time of the proposed control algorithm design on the microcontroller it would be necessary to do a exhaustive (brute force) search for every possible controller input vector $[e\ e_c]$. The input vector is calculated in each sampling interval from the current values of the system set point s and the system state x. The system state makes dependency makes it impossible to change the input vector directly. Therefore, in order to benchmark the control algorithm execution time we conducted a Monte Carlo experiment. For this purpose one hundred random controller setpoints were generated and algorithm execution time was measured for both controllers. This procedure was repeated 100 times with a different seed value for the random generator in each iteration. The results of this experiment are presented in Table II. From those we conclude that the execution time of the proposed controller was almost twice as fast than the traditional PI algorithm, which implies that the proposed approach is superior than the PI controller for control systems where sample time is a critical factor.

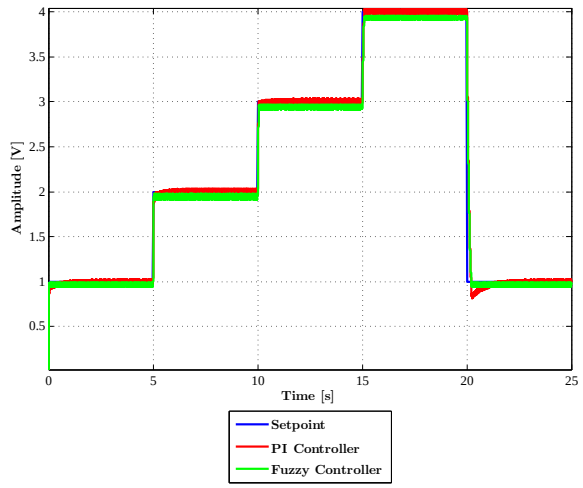

Fig. 11. Simulational results

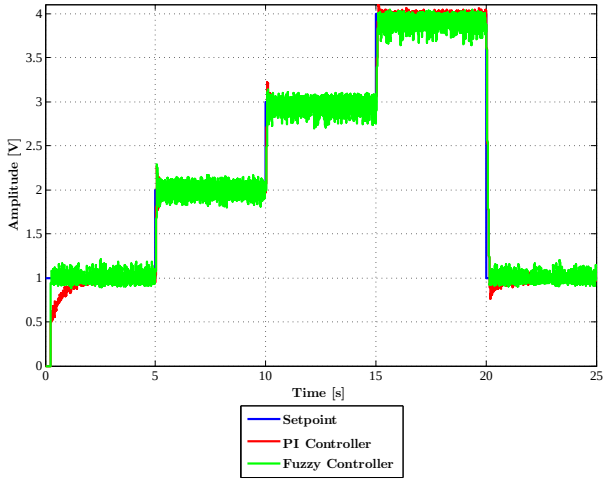

Fig. 12. Experimental results

VI. CONCLUSION

This paper presented an approach to digital controller design based on control surface discretization. The design of a PI controller using this approach was given based on a Fuzzy-PI controller model which behaviour was cloned from a conventional PI controller using the genetic algorithm approach. Controller implementation was done on the Arduino Mega microcontroller platform using a lookup table for control algorithm. Simulation and experimental results demonstrate the useability of the proposed approach. An explanation for the efficiency of this approach can be found in the fact that the onboard memory capacity of modern microcontrollers exceeds their computing power by far. This implies that the proposed lookup table approach is more appropriate for microcontroller based controller design than the traditional computing approach.

Further research will investigate the possibility to solve the following problems:

- Elimination of the statical error by applying this approach an its modifications
- Modular software design on a Microcontroller that can implement any controller without ignoring the hardware restrictions of the specific system.
- Restrictions in controller implementation for nonlinear MIMO systems
- Flexible PC software solution design for rapid controller prototyping

REFERENCES

[1] K. G. Shin & P. Ramanathan, Real-Time Computing: A New Discipline of Computer Science and Engineering, Proceeding of the IEEE, Vol 82, No 1, January 1994.
[2] A. Drumea & P. Svasta, Microcontroller-Based Electronic Module for Controlling Mechatronic Systems, U. P. B. Sci. Bull., Series C, Vol. 71, Iss. 2, 2009.
[3] V. Silva, V. Carvalho, R. M. Vasconcelos & F. Soares, Remote PID control of a DC motor, Manuscript Paper presented at REV2007 conference, Porto, Portugal, June 2007.
[4] J. Chaínho, P. Pereira, S. Rafael & A. J. Pires, A Simple PID Controller with Adaptive Parameter in a dsPIC; Case of Study, 9th Spanish Portuguese Congress on electrical engineering, 2005.

[5] S. S. Gade, A. B. Kanase, S. B. Shendge & M. D. Uplane, Design and Development of Universal On Line Auto Tune PID Controller, International Conference on Control, Communication and Power Engineering, Chennai, Tamil Nadu, India, July 2010.
[6] P. J. Davism, P. Rabinowitz, Methods of Numerical Integration, 2nd ed., Academic Press, 1984.
[7] R. Isermann, Digital Control Systems - Fundamentals, Deterministic Control (Volume 1), 2nd ed., Springer Verlag, 2000.
[8] J. Binfet & B. M. Wilamowski, Microprocessor implementation of fuzzy system and neural networks, in: International Joint Conference on Neural Networks, vol. 1, Washington, DC, 2001.
[9] Y. Tipsuwan, & M. Y. Chow, Fuzzy logic microcontroller implementation for DC motor speed control, IEEE IECON 99, San Jose, CA, 1999.
[10] P. Guillermin, Fuzzy logic applied to motor control, IEEE Trans. on industry applications 32, 1996.
[11] B. G. Hu, G. K. I. Mann & R. G. Gosine, New methodology for analytical and optimal design of fuzzy PID controllers, IEEE Trans. Fuzzy Syst., vol. 7, 1999.
[12] M. Stamp, Once Upon a Time-Memory Tradeoff, 2003.
[13] S. Khan & S. F. Abdulazeez, Design and Implementation of an Optimal Fuzzy Logic Controller Using Genetic Algorithm, Journal of Computer Science 4, 2008.
[14] L. Ljung, Perspectives on System Identification, Annual Reviews in Control, Vol. 34, No. 1, 2010.
[15] The Mathworks, MATLAB - System Identification Toolbox Help, 2010b.
[16] R. C. Dorf, R. H. Bishop, Modern Control Systems, 9th ed., Prentice-Hall, 2001.
[17] L. Ljung, System Identification - Theory For the User, 2nd ed., John Wiley & Sons, 2005.
[18] K. H. Ang, G. C. Y. Chong & Y. Li, PID control system analysis, design, and technology, IEEE Transactions on Control Systems Technology 13, 2005.
[19] R. Fletcher, Practical Methods of Optimization, 2nd ed., John Wiley & Sons, 2005.
[20] X. Tian, X. Wang & Y. Cheng, A Self-tuning Fuzzy Controller for Networked Control Systems, International Journal of Computer Science and Network Security, Vol. 7, No. 1, Jan. 2007.